ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

ORNL/TM-11051
(CESAR-89/01)

# A Concurrent Approach To Model Based Motion Detection

Henrik I. Christensen

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A03; Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-11051
CESAR-89/01

Engineering Physics and Mathematics Division

# A CONCURRENT APPROACH TO MODEL BASED MOTION DETECTION

**Henrik I. Christensen**

Laboratory of Image Analysis
Institute of Electronic Systems
Aalborg University, DK-9000 Aalborg, Denmark

**Published: January 1989**

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE–AC05–84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0289274 2

## TABLE OF CONTENTS

Abstract . . . . . 1

1. INTRODUCTION . . . . . 1

2. MODEL-BASED MOTION DETECTION AND DESCRIPTION . . . . 3

   2.1. Approach . . . . . 3

   2.2. Low-Level Tokens . . . . . 4

   2.3. High-Level Tokens . . . . . 4

   2.4. A System Structure . . . . . 5

3. A CONCURRENT APPROACH TO MOTION DETECTION . . . . . 7

   3.1. Fundamentals . . . . . 7

   3.2. The MPSD Approach . . . . . 8

   3.3. The SPMD Approach . . . . . 9

   3.4. The MPMD Approach . . . . . 10

4. AN IMPLEMENTATION . . . . . 11

   4.1. Token Extraction . . . . . 12

   4.2. Token Matching and Model Correction . . . . . 12

   4.3. Motion Calculation and Estimation . . . . . 13

5. RESULTS . . . . . 15

6. SUMMARY . . . . . 17

Acknowledgments . . . . . 17

References . . . . . 19

Appendix . . . . . 21

# A CONCURRENT APPROACH TO MODEL BASED MOTION DETECTION


## ABSTRACT

In robot-vision it is important to be able to acquire and describe dynamicaly changing scenes. A token based method for real-time motion detection is described here. This method uses a prediction-verfication scheme to cope with ambiguous motion patterns. To improve performance it is investigated how the application may be implemented on a concurrent achitecture like a hypercube. An implementation based on the SPMD approach is presented together with some experimental results. The results demonstrate the systems ability to handle ambiguous situations. Timing results indicate the system has a potential for tracking of less than 10 objects or more than 40 objects, while the advantage of using a concurrent architecture for 10-40 objects is very small.


## 1. INTRODUCTION

Prerequisites for building and operating autonomous systems, such as robots, in a dynamically changing environment are sensor systems that allow the control system to monitor and interact with the world. Sensor for such autonomous systems must be capable of acquiring and analyzing data in real-time. Real-time denotes the analysis of data in a fixed and predefined time frame. This time frame is assumed to be small enough to allow adequate description of phenomena significant to the system. Computer vision has proved to be a very powerful tool in this context. For the description of dynamic phenomena, motion description and analysis tasks are particularly important to a control system. A number of investigators have focused recently on the area of motion detection and description for robotics applications, see for example Andersson [1985], Gonzalez and Wintz [1986], Horn [1986], Aggarwal [1987], Christensen [1987], Sethi and Jain [1987], Waxman et al. [1987], Wiklund and Granlund [1987], Christensen and Granum [1988a; 1988b], Crowley [1988], and Granum and Christensen [1988a; 1988b].

A recent comparison of motion detection techniques [Granum and Christensen, 1988a], concludes that analysis based on correspondence methods seems to offer the best potential for real-time analysis of dynamically changing scenes. The major problem associated with correspondence-based motion description has been the handling of ambiguous situations. As the number of descriptors is often small, the context for matching primitives between images is very limited. However, development of model-based techniques has significantly reduced this ambiguity to a level that allows practical use of this class of methods.

We will outline a model-based motion description approach that was proven to be fast [Christensen and Granum, 1988a] and capable of handling a number of ambiguous situations. The description will also outline a number of techniques for mapping such an application onto a parallel architecture. An implementation on a hypercube computer (NCUBE Corp., Beaverton, OR) will be presented together with encouraging preliminary results. We conclude by summarizing the approach and pointing out issues for future research.

## 2. MODEL-BASED MOTION DETECTION AND DESCRIPTION

### 2.1. Approach

For motion detection based on correspondence or matching methods a number of image descriptors, often referred to as tokens, such as edges, lines or regions are extracted from a sequence of images. Tokens from time-sequential frames are then matched, and based on the difference in location, motion parameters are calculated. The conventional approach to correspondence analysis is shown in Fig. 1.

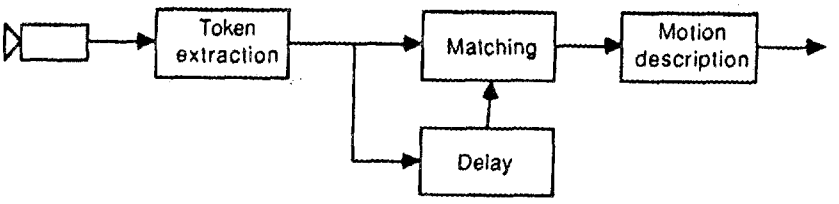


**Fig. 1. A straight forward approach to correspondence analysis.**

In this approach situations can easily arise where the correspondence of tokens is ambiguous, i.e., there might be several matching candidates for tokens. Consider for example the situation in Fig. 2, where a token is approaching the boundary of the field of view, while another enters the field of view.

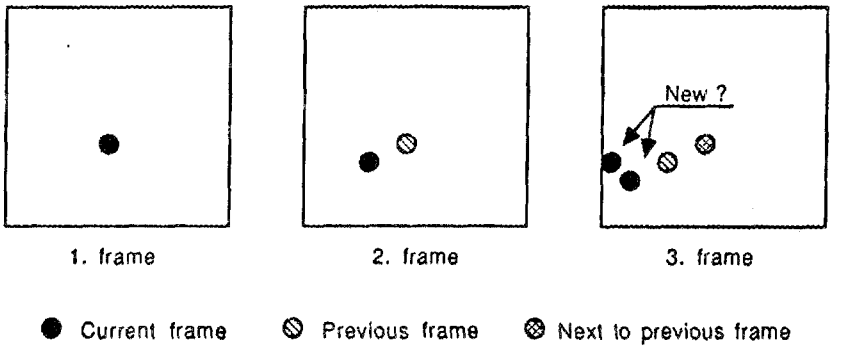


**Fig. 2. A short image sequence with an ambiguous situation.**

If we only consider the last two frames, no information is available that will allow unambiguous matching. If temporal context, i.e., the motion history of tokens, is included, it might be possible to determine an unambiguous correspondence. If it can be assumed that the system is operating continuously, the history of all tokens in the field of view is available for matching. This information might be used in an estimation of token positions in future frames. To do this, the scene history must be accumulated and maintained, i.e.,. a model of the environment must be established. Consider a situation where the scene model contains a statistical description of each token in addition to current position and a number of motion parameters, i.e., velocity and acceleration. Having such information available will enable estimation of the next frame, with subsequent matching between the estimated and the measured frame. This should allow almost unambiguous matching, if it can be assumed that the world is well behaved, i.e., the complexity of scene dynamics is

limited. Motion might be considered well behaved if it is in accordance with the path coherence and smoothness of motion defined by Sethi and Jain [1987] and Jenkins [1983], respectively.

Correspondence type motion detection can be divided into two groups, according to the level of abstraction in token description.

a) Low-level token description
b) High-level token description

## 2.2. Low-Level Tokens

In this category, the primitives describe local image properties. Typical descriptors include edges and line-segments. As the descriptors are local, it is usually possible to extract at large amount of tokens, which facilitates a dense and accurate estimation of image motion.

Low-level tokens are sensitive to noise, as noise can easily introduce edges or break line segments into several smaller ones. This sensitivity introduces additional ambiguity in the matching, i.e. has a token disappeared due to motion or noise? In this category the global image context is unknown, hence it is impossible to handle situations where scene components occlude each other temporarily. Such phenomena must be handled in a postprocessing procedure following motion detection.

## 2.3. High-Level Tokens

These primitives are associated with more global image properties. In this category tokens are typically associated with image regions. The high-level tokens are described by feature vectors that contain statistical and/or structural properties related to the image regions.

To simplify the analysis, primitives may be associated with rigid scene objects. If a nonrigid region is encountered it can be assumed to represent two or more objects currently occluding each other.

The high-level tokens are less sensitive to noise, as noise will have little or no influence on high-level features such as coordinates for the bounding rectangle, area etc. It might have a large influence on features such as the Euler number. Thus, these kinds of features, i.e., topological, should be avoided for the motion detection. The major problem related to motion detection using high-level descriptors is the limited amount of data, i.e. only little contextual information is available for the correspondence analysis, hence it may be more difficult to solve ambiguous situations.

A comparison of low- versus high-level descriptors must take accuracy and ability to handle occlusions into account. In a number of situations the scenario only contains rigid man-made objects, e.g., a factory scene. In such situations it is natural to select the high-level descriptors as they provide a more direct approach to description of scene dynamics. If non-rigid objects may occur, the low-level descriptors seem more appropriate.

## 2.4. A System Structure

Figure 3 shows a block diagram for a model based motion detection system. This structure is applicable to both low- and high-level tokens.

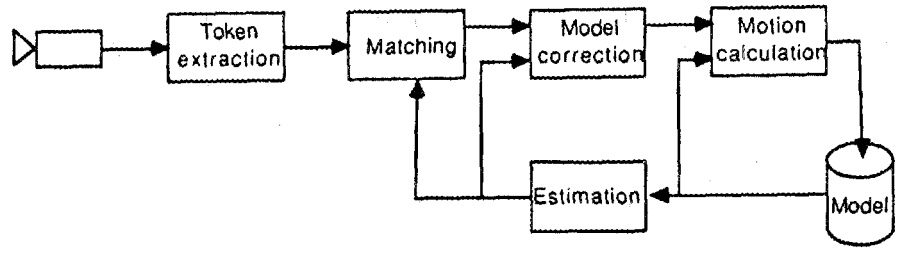


**Fig. 3. Block diagram for a model based motion detection system.**

The token extraction module includes image acquisition, segmentation and feature extraction. The results (a number of feature vectors) are passed on to the matching module. In the motion detection module, the model forms the basis for an estimation of the next frame. The correspondence between estimated and measured frame is subsequently established in the matching module. Matching will not only establish links between corresponding tokens, but will also provide a description of tokens with no correspondence. This information is used to update the estimated model, i.e., objects that have moved into the field of view are introduced and model tokens with no correspondence are discarded. The updated model and the "old" model are finally used for calculation of new motion parameters.

The model-based technique has been used with success by a number of investigators; see for example Aggarwal and Duda [1975], Roach and Aggarwal [1980], Granum and Munk [1986], Christensen [1987], Christensen and Granum [1988a], Crowley [1988], Wiklund and Granlund [1987].

In the following the system described by Munk [1986] and Christensen [1987] will be considered. In this system the scenario is characterized by the following properties:

1) A measured token may correspond to two or more scene items currently occluded.

2) An item (token) in the model represents a single object (edge, line or region) in the measured frame, if and only if it has previously been observed as such.

3) The images have been preprocessed, so that unique primitives (tokens) are available for matching.

4) It is assumed that simple techniques such as global thresholding are sufficient for segmentation, as more sophisticated methods may challenge the real-time requirement. The restriction can be avoided by using special image preprocessing hardware, but special hardware is usually undesirable as it will limit flexibility.

5) For practical reasons, i.e., limited memory, the number of tokens per image is limited to 30. However, the algorithms can easily cope with a large number of tokens.

6) Standard hardware and software should suffice for implementation, and the analysis time should be less than one sampling period. A sampling period is dependent on scene dynamics, i.e., the sampling period might be longer than 30 ms.

The estimation and motion calculation are performed by using second order Taylor expansions of the path. Such an approximation is crude but fast. If the smoothness of motion and path coherence mentioned previously can be applied for a scene this approximation should be sufficient. It has been suggested [Christensen, 1988; Farkas, 1987] that motion description using Kalman filtering might produce more reliable motion descriptors, but the complexity of this method will lower the throughput. Matching is performed by model-based techniques [Munk, 1986] or probabilistic relaxation [Christensen, 1987]. The probabilistic approach has proven to be faster and more general, but in principle both can be applied.

This approach has been implemented in experimental software on standard UNIX computers. Image sequences are preprocessed by the Edinburgh Fast Interval Processor [Shippey et al., 1981], that is capable of producing region descriptions at frame rate (25 Hz). The implementations have been timed and the systems are capable of analyzing 2-5 frames per second, depending on scene complexity.

# 3. A CONCURRENT APPROACH TO MOTION DETECTION

## 3.1. Fundamentals

Christensen and Granum [1988a] reported that such model-based motion detection can cope with ambiguous situations, e.g., occlusions, while a speed of 2-5 frames per second can be obtained on a sequential computer. We describe some approaches to mapping the application onto a parallel architecture. The analysis will focus on the hypercube architecture, but the conclusions should be applicable for a variety of architectures. Figure 4 shows a 4-dimensional hypercube.

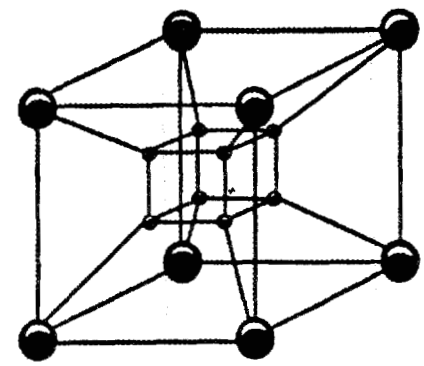

**Fig. 4. A 4-dimensional hypercube configuration.**

The hypercube is a distributed-memory, message-passing architecture, where each processor has local, private memory. Actions are coordinated by messages that are sent on a interconnecting network. The processors (also referred to as nodes) are located at vertices of an n-dimensional cube. An n-dimensional cube can easily be build from 2 (n-1)-dimensional cubes by connecting appropriate nodes, as indicated in Fig. 4. Thus, while the number of processors grows exponentially the number of connections per node only grows linearly. The network is dense enough to allow efficient communication between any two processors, while it is still sparse enough to be simple and inexpensive to implement. In this type of interconnecting network the maximum distance between any two nodes is equal to the dimension of the hypercube. One of the virtues of the architectures is its flexibility. It is possible to map a number of other topologies (rings, grids trees, etc.) onto the architecture. The architecture is thus suited for experiments with concurrent algorithms for a variety of topologies [Heath, 1986].

The hypercube is in principle a Multiple Instruction Multiple Data (MIMD) computer, but due to loosely connected nodes the most common approach to programming is program synchronization rather than instruction synchronization, i.e., the computer is synchronized at a number of milestones, rather than after each instructions. Hence, it is more appropriate to refer to the computer as a Multiple Program Multiple Data (MPMD) architecture.

With this architecture one of the following three processing approaches can be used:

a) execute different programs on each processing element (PE), and communicate data between the PEs; this represents a multiple programs single data (MPSD) approach,

b) distribute data over all PEs and execute the same program on all PEs; this represents a single program multiple data (SPMD) approach, or

c) execute a different program on each PE which processes different types of data, a multiple program multiple data (MPMD) method.

### 3.2. The MPSD Approach

Consider initially the MPSD approach (a). The block diagram in Fig. 3 defines a breakdown of the system into a number of modules with well-defined interfaces. If a delay of three image intervals can be allowed, a structure with one time frame spent on preprocessing, one on matching and one on model correction and motion description, can be developed. In this breakdown different parts of the system are analyzing data from different samples. Thus, it should be simple to map the system on a hypercube architecture, i.e., a PE or a subcube can be assigned to each of the tasks. One possible distribution is shown in Fig. 5.

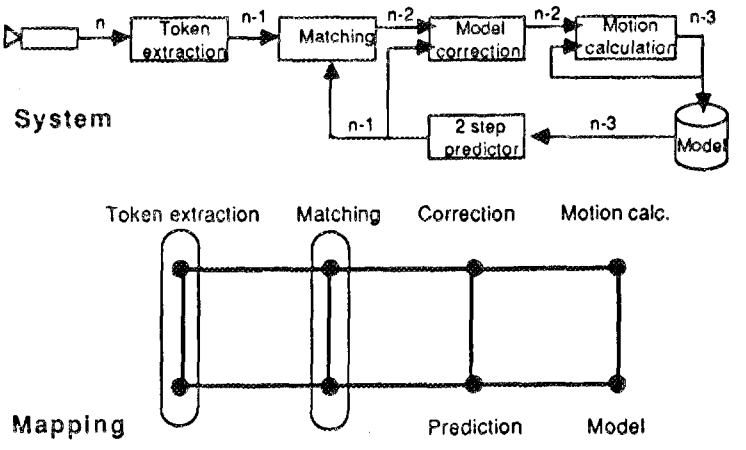


**Fig. 5. Block diagram with a possible mapping onto a 3-dimensional hypercube using a MPSD approach. Two connections for the hypercube configuration have intentionally been left out, as they are not used. The parameter $n$ indicates present time (see text).**

If we determine algorithm complexity for each task, it is apparent that preprocessing will have a complexity that is data-independent, i.e., thresholding, edge detection and/or component labeling will have a constant usage of resources. For the matching it can be shown [Christensen, 1987] that complexity is of the order $O(N^2)$, where $N$ indicates the number of primitives. The other modules will have a complexity of $O(N)$. These numbers indicate that resource usage is data dependent. Hence, it is difficult to assign an appropriate and fixed amount of resources to each of the tasks. If the allocation of resources is performed according to worst-case conditions, a large amount of resources will be idle most of the time. This is particularly true for the

matching part, as it has a second order complexity. A distribution according to mean load will on the other hand imply overload, at least in the matching module, during worst case conditions. This indicates that a balanced distribution of the tasks onto a parallel architecture is very difficult if a MPSD approach is used. Another complication is software design, i.e., the system must be distributed during design, which may be difficult if not impossible. Consider the block diagram in Fig. 5. Here, the matching should be allocated a larger number of processors than the other modules to cope with worst-case conditions. This mapping will be static and applicable only for a narrow range of applications. Thus, this approach will often lead to a nonoptimal use of available resources.

### 3.3. The SPMD Approach

In the SPMD approach all PEs allocated to the system will execute the same program, and the data have been distributed evenly onto the PEs. All modules will thus perform preprocessing, matching, etc., but on different data. The problem is in the distribution of data rather than instructions. For preprocessing this is simple. Consider a situation where each PE is assigned the same amount of data from the input image. In such a situation all PEs should complete the task at approximately the same time. The criterion for data distribution is often minimization of communication between PEs, since communication takes up valuable time that can not be used for computations. For a discussion of image distribution using grid or ring mapping on a hypercube see Jones et al. [1988]. The distribution of data is more complicated for the motion detection, since the use of resources depends upon the input data. The distribution of primitives for motion detection can be performed according to three different approaches.

1) Distribute measured primitives and maintain a complete model of the scene on each PE.

2) Distribute the model and maintain a complete measured frame on each PE.

3) Distribute both scene model and measured primitives on the available PEs.

In approach (1) measured primitives are distributed evenly on the available PEs, and a major portion of the motion detection and description is performed locally at each PE. However, the processing can not be entirely local, e.g., consider a situation where two measured tokens match with the same model token, i.e., an ambiguous situation arises. If the two measured tokens are on different PEs, each of the PEs will interpret this as a single match. To avoid this interpretation the global consistency must be established after the local matching on each PE has been performed. This global analysis requires communication and analysis of the complexity $O(N \log_2 P)$, where $P$ is the number of processors in the hypercube and $N$ is the number of primitives (see also discussion of butterfly algorithm by Jones et al. [1988]). In this approach (1) all the model-tokens are subjected to the same computation on all PEs. If a detailed motion description is required, this will imply redundant calculations that might lower total throughput.

The second approach is similar to that in (1), except that the model is distributed among the processors. This allows for distribution of model updating, estimation, etc., but the problem of global consistency in the correspondence analysis must still

be solved globally. The advantage of this approach with respect to (1) will thus depend on the complexity of model maintenance and estimation tasks.

In the third approach both model and measured primitives are distributed evenly on the available PEs. If the distribution is performed at random, local matching on each PE will probably not be efficient, but if the distribution of tokens is performed according to similarity in geometry, the local matching should be able to establish a major portion of the correspondences. The local correspondences can then be coordinated globally during a postprocessing step. This method should be significantly faster than those in (1) and (2), as both the model and the measured frame have been distributed. Matching, in particular, will be faster as the number of primitives on each PE has been reduced.

The SPMD approach is simple to implement, since only one program, to be executed by all PEs must be developed. Once the data have been distributed the concurrency is hidden to the programmer. We may thus consider the architecture as one computer with a very complex CPU structure.

### 3.4. The MPMD Approach

This type of distribution has the same balancing problems as the MPSD approach. A method is required for balanced distribution of data and algorithm. Considering the problems encountered in MPSD, this method seems to have little to offer for the application at hand.

In summary, it appears, that the SPMD approach is simpler to program and balance, if the algorithms have a nonhomogeneous complexity. In what follows we will only consider the SPMD approach.

## 4. AN IMPLEMENTATION

A model-based motion analysis similar to that in Fig. 3 has been implemented on an NCUBE hypercube. Various aspects of the implementation will be presented here.

The SPMD approach was chosen since it is easier to program and to balance, as explained in Section 3. To optimize with respect to speed, both the measured and model tokens are distributed on the hypercube. The distribution is performed according to image geometry. For preprocessing the image is distributed onto the cube according to a gray code ring mapping, (see Fig. 6). This allows easy balancing of preprocessing.

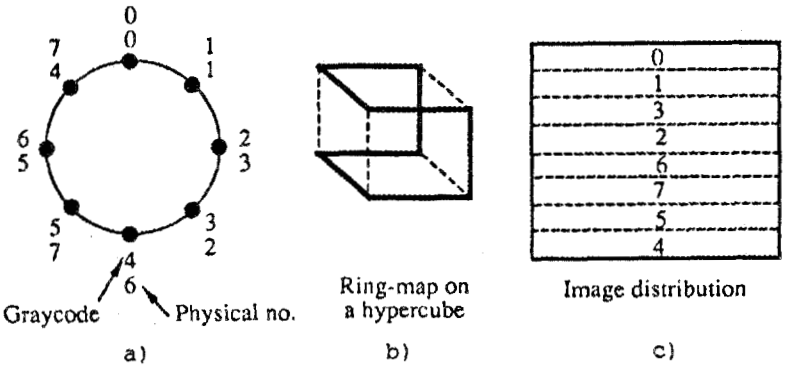


**Fig. 6. Graycode mapping onto a 3-dimensional hypercube.**

a) A ring map with corresponding node numbers,
b) ring mapping on hypercube, and
c) image partitioning with corresponding node numbers.

We will assume that extracted primitives have a uniform spatial distribution in the image. If this requirement is satisfied, the distribution of model and measured tokens can be performed in a manner similar to the ring mapping. The global structure of the program is shown in Fig. 7.

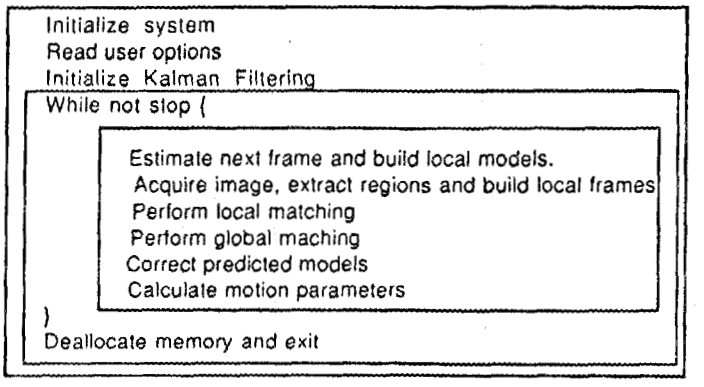


**Fig. 7. Global structure for token matching system.**

The presentation of the individual system tasks has been broken down according to Fig. 3.

### 4.1. Token Extraction

Segmentation and feature extraction from natural scenes is very important for all subsequent analysis. The main objective in the present study has been the development of a concurrent motion detection system. Therefore, a simple and crude preprocessing method has been chosen. Images are acquired and segmented using global thresholding. The threshold is calculated separately for each image based on accumulation of image statistics. Currently the threshold is calculated by the formula

$$T = \text{mean intensity} + 1/10\ (\text{max intensity - min intensity}).$$

It is assumed the background intensities are almost constant and all pixels above the mean intensity belong to scene objects. The approach will obviously fail if the scene is not uniformly illuminated or corrupted by noise, but for experimental purposes it is appropriate.

The segmented image is labeled, and features associated with all foreground regions are calculated. Currently the feature vector contains area of image region, position of centroid, and coordinates for the axis of a bounding rectangle. The preprocessing is performed by function call to the image processing library described by Jones et al. [1988]. A global description covering all image regions is calculated by the functions bin image (thresholding), connect (connected component labeling), and make blob list (feature extraction for labeled image) in the library. Based on the geometry of the distributed image a local frame description is built on each PE for the local matching.

### 4.2. Token Matching and Model Correction

As indicated in Fig. 7, the matching is divided into a local and a global part. Both use a probabilistic relaxation approach similar to that reported by Christensen and Granum [1988a]. The probabilities are updated in each iteration as described by Rosenfeld et al. [1976],

$$p_i^{(n+1)}(\lambda_j) = \frac{p_i^{(n)}(\lambda_j) * S_i^{(n)}(\lambda_j)}{\sum_l p_i^{(n)}(\lambda_l) * S_i^{(n)}(\lambda_l)} \quad .$$

The variable $p_i^{(n)}(l_j)$ is the strength associated with the matching of measured token $t_i$ with model token $l_j$, $n$ indicates the iteration number. S(.) is a support function that indicates the consistency of this labeling with respect to other labelings. The denominator ensures that the sum is normalized. The support function is a simple column-wise weighting. The support function is

$$S_i^{(n)}(\lambda_j) = \frac{p_i^{(n)}(\lambda_j)}{\sum_k p_k^{(n)}(\lambda_j)} \quad .$$

This support function has proven to be able to handle ambiguous situations such as occlusions, and it does not incorporate any domain knowledge. Hence it is generally applicable. For a more detailed discussion of the matching function see Christensen and Granum [1988a].

After all local matching tasks have converged, a global matching table is built and all the measured or estimated tokens that have not been matched are included in a global matching, similar to the local one. The combined matching result is finally interpreted, i.e., labels such as SINGLE, STARTER, OCCLUSION, ALONE, etc. are assigned to the model tokens and new tokens are introduced. If a model token has no corresponding token in the measured frame, a flag is set and a variable (age) is incremented. If such a token has been without a matching partner in three consecutive frames, it is discarded from the model. The delayed rejection of model tokens without partners has mainly been incorporated to handle situations with noise. A token may disappear for one or two frames without any loss of tracking.

### 4.3. Motion Calculation and Estimation

Compared to the implementations by Munk [1986] and Christensen [1987], this implementation has more computational resources available, hence it was decided to change the motion calculation from a second order Taylor expansion to a Kalman filtering. The Kalman filtering is more robust to noise, as it incorporates a larger amount of the temporal context and the implied estimate of covariance of the prediction error might be incorporated into the matching. This motion description will also allow a minimum squared error estimation, while the Taylor expansion in principle may result in an unbounded estimation error. A second order model was chosen for the motion calculation, allowing for calculation of both velocity and acceleration. The prediction is a simple second order Taylor expansion based on the motion parameters in the model.

The current model does not incorporate a mechanism for handling sudden changes in motion, i.e., it assumes motion can be modeled by a time-invariant system. The Kalman filter should thus be changed at a later stage to handle motion models corresponding to time-varying systems. This can be achieved in these three ways,

a) variable covariance methods,
b) recursive system identification, and
c) multiple model Kalman filtering.

In variable covariance methods the detected estimation error is compared to an estimate of the error. If the error becomes too large, the estimate of the covariance of the prediction error is increased to cope with a possible maneuver. This method has the advantage of being simple, but it can only handle small changes. If major changes occur, the method will fail, i.e., the Kalman filter is only suited identification of systems with well-behaved dynamics.

In recursive system identification, the Kalman filtering is replaced with a model that has time-varying parameters. The parameters (velocity and acceleration) are then estimated recursively. In traditional control theory this is accomplished by calculating a covariance matrix for the estimation error, rather than modelling it. Such an approach will automatically adapt to time-varying models, but it is far more complex in terms of computational resources.

In the multiple model filtering, the motion calculation is based on a number of models of different order. The model with the smallest estimation error is then selected for current processing. In stationary mode a low order model will be used, while sudden changes in the motion will result in use of higher order models. The order of the model will thus change in response to the detected motion, i.e., we have a model adaptive approach. For a more complete description of methods to estimate system parameters refer to conventional control system literature, such as Aastrm and Wittenmark [1984] and Bar-Shalom and Fortmann [1988].

## 5. RESULTS

The implemented system has been tested on a number of sequences of synthetic images. To demonstrate the potential of the system, results from a few of these sequences will be presented. The system has not been tested on natural images, as the segmentation poses serious problems in handling noise and nonuniform illumination. For application to natural scenes this part of the system should be redesigned.

The first sequence contains three objects. An object is moving in a semicircle in the bottom part of the image while the second moves a in semicircle in the upper part. A third object moves across the middle of the field of view with constant velocity. In the middle of the field of view all three object meet and occlude each other. From the plot in Fig. 8, it can be seen that the system is capable of tracking the objects. It can also resolve the ambiguous situation in the middle of the sequence. Minor deviations from the true paths can be detected, but that is to be expected. The lines represent a trace of the centroid for the estimated objects (i.e., the system interpretation), while the rectangles represent an approximate outline of measured objects.

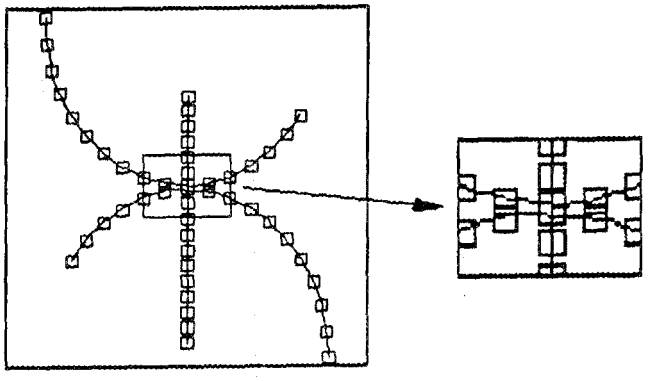

**Fig. 8. A test sequence with three objects that occlude each other in the middle of the field of view.**

The next test sequence shows two objects that occlude each other initially, but as the sequence progresses, they move apart. The plot in Fig. 9 shows how the separate trajectories are adapted as soon as the separate objects are detected. This indicates that the system is capable of handling a number of ambiguous situations.

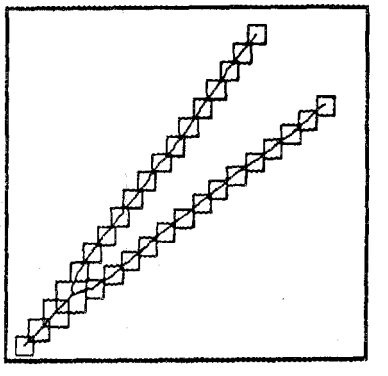

**Fig. 9. A sequence where two objects initially occlude each other.**

The implementation on the hypercube has been tested on a number of test sequences, including those in Figs. 8 and 9. Preprocessing of the images takes approximately 2.1 seconds, while token tracking involving less than 10 tokens can be achieved in 0.3-0.4 seconds. All measurements were performed on a 4-dimensional hypercube. If the dimension of the hypercube is increased, the preprocessing will take less time, the speed-up being almost linear. Motion analysis on the other hand does not improve significantly, mainly because the number of tokens is smaller than the number of processing elements. The results for sequences containing 20, 30, and 40 tokens are shown in Table 1.

**Table 1. Timing of motion detection system for a different number of tokens on a selected range of hypercube dimensions.***

| Dimension | Preprocessing | 20 tokens | 30 tokens | 40 tokens |
|---|---|---|---|---|
| 3 | 4.08 | 0.63 | 1.10 | 1.36 |
| 4 | 2.12 | 0.59 | 0.93 | 0.94 |
| 5 | 1.27 | 0.52 | 0.81 | 0.77 |
| 6 | 1.05 | 0.54 | 0.73 | 0.68 |

*All numbers are in seconds. The timing was performed on a 6 MHz NCUBE Corp. computer.

The processing speeds obtained with this implementation are shown in Fig. 10.

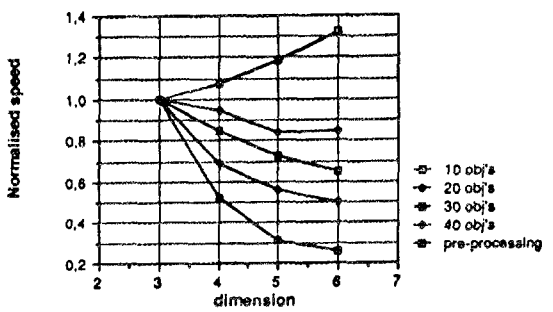


**Fig. 10. Normalized speed-up factors obtained with the motion detection system on an NCUBE computer.**

Figure 10 shows nonlinear speedup. It appears that speedup factors deteriorate as soon as we have more PEs than tokens; as could be expected. The timing results indicate the system is suited for tracking of a few objects (less than 10) on a low dimension hypercube or tracking of a large amount of tokens on a high dimension cube. The system seems to have little potential for tracking of 10-40 tokens, since speed-up factors in this range are poor. It may be possible to obtain better results if another distribution method is applied. Jones [1988] has suggested an approach where the entire measured frame is kept on all PEs, while only the model is distributed. This will probably balance better, but due to second order algorithm complexity in the matching, the increase in speed will probably be insignificant.

## 6. SUMMARY

A model-based approach to motion detection has been outlined and strategies for mapping the application onto a hypercube architecture have been discussed. A practical implementation was briefly sketched together with some results, indicating that the system performs well in ambiguous situations. A processing speed on the order of 2-4 frames/second for sequences with a limited number of tokens (10) was also reported. The major weakness in the system is the preprocessing, which takes 10-20 times longer than the actual motion detection. The segmentation in the preprocessing is also sensitive to noise and nonuniform illumination. Additional work in this area should thus focus on preprocessing.

For practical applications one might use special hardware for the preprocessing. The PIPE [Kent et al., 1985] and the Cytocomputers [Lougheed and McCubbrey, 1985] are such computers, where the images are preprocessed in a pipeline of 1 to 10 processors. The use of special preprocessing hardware will, however, limit the flexibility, but for most applications that might be acceptable.

### Acknowledgements

A number of people have made this research possible. I would like to express my gratitude to a few of them. I thank E. Granum, Aalborg University, and R. C. Mann, Oak Ridge National Laboratory (ORNL), who made my visit to ORNL possible. I would also like to thank E. Granum for motivating and inspiring much of my work in motion detection and analysis. I also thank J. P. Jones, ORNL, who took the time for a number of discussions on vision in general and concurrent computer vision in particular. L. Farkas offered his help in the introduction to the NCUBE environment at ORNL, and he provided a large portion of a matrix library.

The funding provided by the Danish Technical Research Council, under the FTU-programme, grant no. 5.17.5.6.06, the Danish Research Academy's December 1987 traveling scholarship, U.S. Department of Energy, under contract DOE-AC05-840R214, and the equipment from the Foundation Vision North, Aalborg is gratefully acknowledged.

## References


1. Aastrm, K. J. and Wittenmark, B., *Computer Controlled Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.
2. Aggarwal, J. K., "On the Computation of Motion from a Sequence of Monocularor Stereo Images–An Overview," in *Machine Intelligence and Knowledge Engineering for Robotics Applications*, (Eds.) A. K. C. Wong and A. Pugh, NATOASI series F, **33**, Springer Verlag, New York, 1987.
3. Aggarwal, J. K. and Duda, R. O., "Computer Analysis of Moving Polygonal Images," *IEEE Trans. on Comput*, 966-976, October 1975.
4. Andersson, R. L., "Real Time Gray Scale Video Processing Using a Moment Generating Chip," *IEEE-RA-1*, **2**, June 1985.
5. Bar-Shalom, Y. and Fortmann, T. E., *Target Tracking and Data Association*, Academic-Press, Boston, MA, 1988.
6. Christensen, H. I., "Monitoring Moving Objects in Real-Time," M.Sc. Thesis, Institute of Electronic Systems, Aalborg University, Denmark, June 1987.
7. Christensen, H. I., "On Motion Description and Estimation," *BAG-MEMO-8801*, Laboratory of Image Analysis, Aalborg University, Denmark, January 1988.
8. Christensen, H. I., and Granum, E., "On Token Matching for Real-Time Motion Analysis," Lecture Notes in Computer Science, **301**, (ed.) J. Kittler, Springer Verlag, New York, 448-457, March 1988.
9. Christensen, H. I., and Granum, E., "On Multi-Scale Motion Analysis," *Procs. NOBIM konferensen 1988*, Norske Regnesentralen, Olso, Norway, June 1988.
10. Crowley, J. L., Stelmaszyk, P. and Discours, C., "Measuring Image Flow by Tracking Edge-Lines," *2nd ICCV*, Tarpon Springs, Fla, December 1988.
11. Farkas, L., "Tracking a Moving Object in a Sequence of TV-images Using a Hypercube Concurrent Computer," CESAR-87/54 report, Oak Ridge National Laboratory, Oak Ridge, TN, 1987.
12. Gonzalez, R. C., and Wintz, P., "Digital Image Processing," Addison Wesley, NY, 1986.
13. Granum, E. and Christensen, H. I., "Methods for Real-Time Motion Analysis," *Proc. 2nd Hungarian Work Shop on Image Processing,* Budapest, June 1988.
14. Granum, E. and Christensen, H. I., "On Principles of Motion Analysis in Real-Time," *Proc. 8th SPIE Topical Conf. 1027*, Hamburg, September 1988.
15. Granum, E. and Munk, K. H., "Monitoring Moving Objects in Real-Time," *BAG-MEMO-8601*, Laboratory of Image Analysis, Aalborg University, Denmark, March 1986.
16. Heath, M. T., "The Hypercube: A Tutorial Overview," *Proc. 1986 Conf. on Hypercube Comp.*, M. T. Heath (Ed.), *SIAM*, pp. 7-11, 1986.
17. Horn, B. P. K, "Robot Vision," McGraw-Hill, NY, 1986.
18. Jenkins, M., "Tracking Three Dimensional Moving Light Displays," *Proc. Workshop Motion Representation*, Toronto, Ont., Canada, pp. 66-70, 1983.

19. Jones, J. P., Personal communication, Engineering Physics and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, September 1988.

20. Jones, J. P., R. C. Mann and E. M. Simpson, "A Computer Vision System for a Hypercube Concurrent Ensemble," *ORNL/TM-10679*, Oak Ridge National Laboratory, Oak Ridge, TN, August 1988.

21. Kent, E. W., Shneier, M., and Lumia, R., "PIPE - Pipelined Image Processing Engine," *Jour. Parallel and Distributed Computing*, **2**, pp. 50-78, 1985.

22. Lougheed, R. M. and McCubbrey, D. L., "Multi-processor Architectures for Machine Vision and Image Analysis," *Proc. International Conf. on Parallel Processing*, pp. 493-497, 1985.

23. Munk, K. H., "Bevaegelsesanalyse ved brug af hojniveau billedrepraesentation," M.Sc. Thesis (in Danish), Institute of Electronic Systems, Aalborg University, Denmark, June 1986.

24. Roach, J. L. and Aggarwal, J. K., "Determination of Movement of Objects from a Sequence of Images," *IEEE-PAMI-2*, pp. 554-562, November 1980.

25. Rosenfeld, A., Hummel, R. A. and Zucker, S., "Scene Labelling by Relaxation Operators," *IEEE SMC-6*, **6**, pp. 421-428, June 1976.

26. Sethi, I. K. and Jain, R., "Finding Trajectories of Feature Points in a Monocular Image Sequence," *IEEE PAMI-9*, **1**, pp. 56-58, January 1987.

27. Shippey, G. A., Bailey, R. J. H., Farrow, A. S. J., Rutovitz, D., Tucker, J. H., "A Fast Interval Processor," *Pattern Recognition*, **14, 1–6**, pp. 345-365, 1981.

28. Waxman, A. M., LeMoigne, J. J., Davis, L. S., Srinivasan, B., Kushner, T. R., Liang, L., Siddalingaiah, T., "A Visual Navigation System for Autonomous Land Vehicles," *IEEE RA-3*, **2**, June 1987.

29. Wiklund, J. and Granlund, G., "Image Sequence Analysis for Object Tracking," *Proc. 5th Scandinavian Conf. on Image Analysis*, Stockholm, June 1987.

## Appendix

The source code for the system described in this memo is available on the CESAR/NCUBE computer at Engineering Physics and Mathematics Division, Oak Ridge National Laboratory. The source code and executables are located in the directory: //orn10/usr/hic/src/matching, the headerfile for the system is in //orn10/usr/hic/src/include. Each of the source code files is briefly described below.

| | |
|---|---|
| tokentrack.h | Headerfile; it contains an external definition of all global variables, the basic data-structure for the dynamic scene model and forward declaration of functions. |
| main.c | The main structure of the token matching program and a definition of all global variables. |
| init.c | Functions for initialization of system and decoding of user options. |
| regextract.c | Preprocessing routines for image acquisition, segmentation and feature extraction. |
| model.c | Model correction,estimation and motion calculation functons. |
| kalman.c | The Kalman filter. |
| kal_3lib.c | Functions for initializing the Kalman filter. |
| match.c | Source code for the probabilistic relaxation and interpretation of matching results. |
| mat.c | Source code for matrix library. This file includes a number of matrix function implemented by L. Farkas, ORNL (Spring 1988). |
| tokentrk | Compiled node program for token tracking. |
| genseq.c | Generator for synthetic image sequences. |
| genseq | Compiled node program for syntehtic image sequence generation. |

A small demo version of the system is located in the directory //orn10/usr/hic/demo/track, a single sequence of images is contained in the files seq1.[0-19], while the file trkgo contains a script file that allows demonstration of the system.

## INTERNAL DISTRIBUTION

1. B. R. Appleton
2. M. Beckerman
3. J. C. Culioli
4. J. Han
5. J. P. Jones
6. C. W. Glover
7–11. R. C. Mann
12. F. C. Maienschein
13. E. M. Oblow
14. F. G. Pin
15. C. R. Weisbin
16. B. A. Worley
17. J. J. Dorning (consultant)
18. R. Haralick (consultant)
19. EPMD Reports Office
20. Central Research Library
21. ORNL Technical Library
22–23. Laboratory Records
24. ORNL Patent Office
25. Laboratory Records–RC

## EXTERNAL DISTRIBUTION

26–30. H. I. Christensen, Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University, DK-9000 Aalborg, Denmark

31. Office of Assistant Manager, Energy Research and Development, Oak Ridge Operations Office, US/DOE, P.O. Box 2001, Oak Ridge, TN 37831

32–41. Office of Scientific and Technical Information, US/DOE, Oak Ridge, TN 37831