TR－H－008

Implementation of Feature Tracking and Factorization Algorithm for Shape and Motion Recovery from Image Streams

Maarten W. Borggreven
Shinjiro KAWATO

1993. 4. 9

ATR人間情報通信研究所

〒619-02 京都府相楽郡精華町光台2-2　☎07749-5-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1011
Facsimile: +81-7749-5-1008

Implementation of Feature Tracking and Factorization Algorithm for Shape and Motion Recovery from Image Streams

Maarten W. Borggreven

April 1993

ATR Human Information Processing Labs
Hikaridai, Seika-cho, Soraku-gun
Kyoto, 619-02 Japan

Abstract

In 1990 Tomasi & Kanada started a series on Shape and Motion Recovery from Image Streams under orthographic projections. This report contains a first implementation and analysis of the main algorithms in order to provide a solid basis for own research on this topic. First aspect is the selection of feature points, or rather feature windows. Selection is done on basis of thresholding eigenvalues of the coefficient matrix, containing the gradient vectors of the feature window. Second aspect is the tracking of the feature windows along the image stream. Final aspect is the actual 3D shape recovery using a proposed factorization method. Singular Value Decomposition is used to decompose the measurement matrix into shape and motion. The method further calls upon the rank principle for noisy images to remove redundancy from the measurements. A few experiments are done to analyse the performance and create thoughts about future research.

1. Introduction

For many years algorithms are presented to recover 3D shape and camera motion from multiple 2D images. Traditional methods are mainly based on computing differences between depth values. Usually assumptions are made so no general coverage is possible. The algorithm presented in the **Shape and Motion from Images** paper series of Tomasi & Kanade, part II [Tomasi & Kanade,1991a] and part III [Tomasi & Kanade,1991b] heads towards another direction, with less limitations, and forms the basis of our experiment. The method assumes orthographic projection, so the component along the optical axis can not be computed. However, this restriction enables direct decomposition of the image stream into object shape and camera motion, without computing depth as an intermediate step.

In part III of the series, *Detection and Tracking of Point Features*, a method based on the calculation of eigenvalues of gradient matrices is discussed to select 'trackable' feature points (actually feature *windows*) and track them along an image stream. The image stream represents long sequence of images covering a wide motion in small steps. The method uses this large number of frames and feature points to reduce the noise sensitiviness. The P tracked feature points of the F images can be represented by a measurement matrix, containing the horizontal and vertical coordinates.

In part II of the series, *Point Features in 3D Motion*, Tomasi & Kanada propose a factorization method of the measurement matrix based on Singular Value Decomposition, simplifying the computation distinguishly. The Rank Theorem, implying that the measurement matrix under ortography is of rank 3, justifies partitioning the matrix into a $2F \times 3$ motion matrix and a $3 \times P$ shape matrix. Under noisy conditions, these are the best possible estimates for the true matrices. Good results are obtained to achieve recovery of 3D shape as well as camera motion. It is actually in this area of the field where most further research should be done. So that is where our experiments will be heading towards in the end. However, before adding new ideas the proposed algorithms should be implemented in software so we can create a good environment and setting for the experiments.

2. Theory

The steps followed to implement the algorithms presented by Tomasi & Kanada are outlined in this chapter. The method assumes ortographic projection and the two reference systems are illustrated in Fig.2.1 . The basic steps are listed below and later described in further detail in the sections of the chapter.

- **Data Acquisition** : the frames of the image stream are stored in an easily accessable way, using camera, frame grabbing software and a real time disk storage device.

- **Feature Point Selection** : actually this means selection of feature *windows* containing sufficient texture, since pixels are difficult to track. The windows are selected by a procedure thresholding the minor eigenvalues of the gradient matrix of the windows.

- **Feature Point Tracking** : the feature windows are tracked along the image stream, calculating the displacement vectors. Windows that suffer of too much variation along the sequence or can not be tracked anymore due to occlusion are omitted. The remaining vectors are combined to the measurement matrix.

- **Factorization Method** : the measurement matrix is decomposed into singular values. The resulting matrices are partitioned, based on the rank principle, implying the matrix to be highly redundant. From the partitioned decomposition of the measurement matrix, the 3D shape of the object as well as the camera motion is now easily computed.

- **Alignment of Camera Reference System** : a rotation matrix is calculated to align the camera reference system to the world reference system.

It is meant to subject especially the 3D recovery part to change and implement our own improved algorithm. Since only vague ideas have come up in this early stage, first the existing algorithm will be qualitatively investigated by some performance analysis.

2.1 Data Acquisition

- Grab and digitize snapshots from the scene, while moving the camera.
- Store the succeeding frames into a Real Time Disk device.
- Transport the image sequence from RTD into UNIX files, registered by an indexnumber. Each file now contains a single frame of the image sequence, ready to be subjected to the actual computations.

2.2 Feature Point Selection

- Determine a window size large enough to minimize the aperture problem, causing sensitiviness to noise and small enough to minimize the occlusion problem.
- Estimate the gradient vector $\vec{g} = (\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y})$ for all the image points in the starting frame.
- Compute the symmetric 2×2 coefficient matrix

$$\mathbf{G} = \int_{\mathcal{W}} \vec{g}\,\vec{g}^T\, w\, dA$$

with w a weighting function, for *all* possible windows $\mathcal{W}$ and calculate their eigenvalues λ_1 and λ_2.

- Accept the windows $\mathcal{W}$ for which $min\,(\lambda_1, \lambda_2) > \lambda_{th}$ is true, with λ_{th} being a predefined threshold. These windows serve as a tool to track the feature points.

2.3 Feature Point Tracking

- Estimate the gradient vector $\vec{g}$ for all the feature points in all frames.
- Compute for all feature points the difference $\vec{h} = I(x, y, t) - I(x, y, t + \Delta t)$ between all pairs of succeeding frames.
- Calculate for all feature windows in all pairs of succeeding frames the displacement vector $\vec{d} = (\Delta x, \Delta y)$ which minimizes the weighted residu

$$\epsilon = \int_{\mathcal{W}} (\vec{h} - \vec{g}\vec{d})\, w\, dA$$

- If a predefined threshold $\Delta\epsilon_{th}$ can not be reached, bilinearly interpolate the window $\mathcal{W}$ with vector $\vec{d}$ resulting in a new vector $\vec{h}$ and repeat the last step.

- Reject the feature windows that do not reach the threshold or are occluded somewhere in the image stream.

- Calculate with the displacement vector $\vec{d}$ and the initial coordinates for the remaining set of feature points a sequence of coordinates $\{(u'_{fp}, v'_{fp}) | f = 1, ..., F \wedge p = 1, ..., P\}$.

2.4 Factorization Method

The method of decomposition of the measurement matrix and the recovery of shape and motion described in this section is called the Factorization Method, presented by Tomasi & Kanada in their second paper of the **Shape and Motion from Image Streams** series [Tomasi & Kanada, 1991a].

2.4.1 Decomposition of the Measurement Matrix

- Register the coordinates by substracting the average over the total of feature points :

$$u_{fp} = u'_{fp} - \frac{1}{p} \sum_{p=1}^{P} u'_{fp}$$

$$v_{fp} = v'_{fp} - \frac{1}{p} \sum_{p=1}^{P} v'_{fp}$$

- Create the $2F \times P$ measurement matrix $\mathbf{W} = \left[\frac{\mathbf{U}}{\mathbf{V}}\right]$ with $\mathbf{U}$ and $\mathbf{V}$ containing the coordinate pairs (u_{fp}, v_{fp}).

- Decompose the measurement matrix, applying the Singular Value Decomposition (SVD) method, into $\mathbf{W} = \mathbf{L} \cdot \mathbf{\Sigma} \cdot \mathbf{R}$ with $\mathbf{\Sigma}$ containing the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_P$ on its diagonal.

2.4.2 Recovery of 3D Shape and Camera Motion

- Assuming $2F \geq P$ the matrices are now partioned as follows :

$$\mathbf{L} = [\overbrace{\mathbf{L}'}^{3} \mid \overbrace{\mathbf{L}''}^{P-3}]$$

$$\mathbf{\Sigma} = \left[\begin{array}{c|c} \overbrace{\mathbf{\Sigma}'}^{3} & \overbrace{0}^{P-3} \\ \hline 0 & \mathbf{\Sigma}'' \end{array}\right]$$

$$\mathbf{R} = \left[\begin{array}{c} \overbrace{\mathbf{R}'}^{P} \\ \hline \mathbf{R}'' \end{array}\right]$$

so that $\mathbf{W} = \mathbf{L}' \cdot \mathbf{\Sigma}' \cdot \mathbf{R}' + \mathbf{L}'' \cdot \mathbf{\Sigma}'' \cdot \mathbf{R}''$. If $2F \leq P$ partition the transpose of $\mathbf{W}$.

According to the Rank Principle for noisy measurements, the most significant shape and motion information in $\mathbf{W}$ is contained in its three greatest singular values, together with the corresponding left and right matrices. So the best estimate for the ideal (no noise) situation is matrix $\hat{\mathbf{W}} = \mathbf{L}' \cdot \mathbf{\Sigma}' \cdot \mathbf{R}'$.

- The desired motion and shape matrices can therefore be computed from :

$$\mathbf{M} = \hat{\mathbf{M}} \cdot \mathbf{A}$$
$$\mathbf{S} = \mathbf{A}^{-1} \cdot \hat{\mathbf{S}}$$

with $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ being linear transformations of the true motion and shape matrices computed from :

$$\hat{\mathbf{M}} = \mathbf{L}' \cdot [\mathbf{\Sigma}']^{\frac{1}{2}}$$
$$\hat{\mathbf{S}} = [\mathbf{\Sigma}']^{\frac{1}{2}} \cdot \mathbf{R}'$$

and with $\mathbf{A}$ being any 3×3 invertible matrix that solves the quadratic system :

$$\vec{i}_f \cdot \mathbf{A} \cdot \mathbf{A}^T \cdot \vec{i}_f = 1$$
$$\vec{j}_f \cdot \mathbf{A} \cdot \mathbf{A}^T \cdot \vec{j}_f = 1$$
$$\vec{i}_f \cdot \mathbf{A} \cdot \mathbf{A}^T \cdot \vec{j}_f = 0$$

This system is yielded by two *metric constraints* which state that the rows of matrix $\mathbf{M}$ are unit vectors and that the first F are orthogonal to the corresponding F in the second half.

Note however that these decompositions are not unique, since the equations are overconstrained. So, if $\mathbf{A}$ is an invertible 3×3 matrix, the matrices $\hat{\mathbf{M}}\mathbf{A}$ and $\mathbf{A}^{-1}\hat{\mathbf{S}}$ are also solutions to this system, since

$$(\hat{\mathbf{M}}\mathbf{A})(\mathbf{A}^{-1}\hat{\mathbf{S}}) = \hat{\mathbf{M}}(\mathbf{A}\mathbf{A}^{-1})\hat{\mathbf{S}} = \hat{\mathbf{M}}\hat{\mathbf{S}} = \hat{\mathbf{W}}.$$

Thus, $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ are, except for noise, linear transformations of the true motion and shape matrices $\mathbf{M}$ and $\mathbf{S}$. Geometrically this means that the solution is determined up to a rotation, since the orientation of the camera with respect to the world reference system is arbitrary. In the next section this alignment problem will be briefly discussed.

2.5 Alignment of the Camera Reference System

Since the solution is by now determined up to a rotation the last step to be solved is the alignment of the recovered 3D shape to the world reference system.

- Find the rotation matrix $\mathbf{R}$ which minimizes the residue

$$\left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \mathbf{R} \cdot \begin{bmatrix} \vec{i}_1 & \vec{j}_1 & \vec{k}_1 \end{bmatrix} \right\|$$

where $\vec{i}_1$ and $\vec{j}_1$ are the first and $F+1$ -th row of matrix $\mathbf{M}$, and $\vec{k}_1 = \vec{i}_1 \times \vec{j}_1$. Since the column vector of the axes matrix are orthogonal, $\mathbf{R} = [\vec{i}_1 \quad \vec{j}_1 \quad \vec{k}_1]^T$.

- Multiply the matrices $\mathbf{M}$ and $\mathbf{S}$ with the rotation matrix $\mathbf{R}$ to get the final aligned 3D information.

3. Implementation

In this chapter an overview is given of what has been accomplished and implemented in software so far. The steps outlined in Tomasi & Kanada are first implemented in its original state to get an idea of its performance. Later efforts will be made to improve and enhance their technique of recovering shape and motion.

Programs are written in standard C language, free to be accessed and modified by anyone. A total of five programs are made to achieve the implementation of the complete theory. In the next sections the program's input and output are briefly discribed and for convenience the input and output files are listed in Appendix A. First, an introduction to ***Visilog*** will be given, the software package we used to grab the images.

3.1 Visilog

The ***Visilog*** software package, distributed by NOESIS Software Inc. France, is a full image processing package. We only used a small part of its possibilities, mainly Real Time Disk (RTD) applications. ***Visilog*** consists of two seperate tasks, the **Image Processing Engine (IPE)** and the **Resource Manager (RM)**. As the name already implies, the **IPE** takes care of all image processing tasks. The **RM** is for managing the graphical interface, interpreting user commands and C scriptfiles. The organization is shown in Fig.3.1 . The C interpreter with the help of precoded processing functions, allows you to easily write scriptfiles which perform image processing tasks.

A scriptfile has been develloped with the ***Visilog*** software to achieve a convenient interface to grab and store the image sequence. Since the RTD services animations purposes, but computations are done on the workstation processors, easy transportation of the image frames from RTD into UNIX files and viceversa should be made possible. The scriptfile provides these needs. For convenient access, ***Visilog*** also provides the possibility to attach your own processing function to its main menu.

3.2 Feature Window Selection

To select the feature points/windows from the first frame, two C programs are written. The first reads the specified starting frame from disk and calculates the minor eigenvalues λ for all possible windows in this first frame. The list of eigenvalues is then sorted and stored to make an easy selection possible. The second program actually selects the eigenvalues on basis of a threshold, resulting in a set of good 'trackable' feature windows. Unlike Tomasi & Kanada stated we could not pick one threshold that holds for different types of scenes.

3.3 Feature Window Tracking

The tracking of the feature windows along the image stream is performed by a third program, calculating the displacement vector between the succeeding frames. As input serve the selected feature windows of the first frame. The result is a file containing the displacements of all windows during the sequence. A fourth program draws the tracked windows as white squares onto the original image frame files, which makes easy perfomance analysis possible.

3.4 Shape Recovery

The last program performs the actual shape and motion recovery by factorizing the measurement matrix. As described in the preceding chapter, the displacementvectors, are first registered in a measurement matrix. In the program IMSL C/Math library functions are used to factorize, decompose and multiply matrices. Results are final files containing the aligned 3D shape and motion information. Any 3D sofware can be used to view the results.

4. Performance Analysis

In the first stage of this research it is necessary to analyse the proposed theory to form an idea of its performance. In a later stage new ideas can be added to the promising proposals. This chapter discusses the theory in a little more detail and gives some results of the test experiments we carried out. Also typical difficulties as occlusion and shiny surfaces are briefly touched upon here. We mainly used two similar image scenes in our experiments : MILKPACK and CARD. The begin and ending image (25 frames apart) of both series are shown in Fig.4.1 . The objects are kept simple, so easy 3D verification is possible.

4.1 Setup

First of all a few words about the setup we used for the experiments. Since either the camera or object has to move to produce an image stream, we chose the more convenient one : object movement. To simplify the setup, the objects used in the experiments are put on a software controlled rotation platform. This setup is sufficient for the first analysis of the algorithm. When it is clear what precise part will be enhanced or improved, a more specific and refined experimental setup can be made.

When the rotation angle between successive frames is small enough, it satisfies the condition to call it a proper image stream. That is, the rotation should produce a displacement fine enough not to loose information. Typically, this displacement requirement results in a maximum of only a few pixels between successive frames.

4.2 Feature Selection

As already mentioned, good features are points that can be tracked well. However since points can not be easily tracked, especially in noisy images, it is better to use windows. Obviously the windows must satisfy some conditions as will be pointed out in the next sections.

4.2.1 Window Size

The size of the feature windows will not be discussed in detail here. Tomasi & Kanade discussed this in his paper sufficiently enough to only give the results. Simply stated, a small window size raises problems because of noise sensitiviness and a large window size gives rise to too much occlusion. Tomasi & Kanade used a window size of 15×15 pixels, which yields good results for most cases. However, he advices to devellop automatic window size selector, since this can improve the perfomance of the algorithm. This is one subject we want to investigate in more detail, but is not implemented yet at this stage.

4.2.2 Overlap

It is quite logical that in many cases the possible feature windows are concentrated in particular areas, like fuzzy ones, and overlap eachother. This is not a very good basis to form an idea of the 3D structure of the whole image scene. More important however are the severe problems distortion causes when rotating the camera. Close windows will even be more dense after a few rotation steps and proper distinction becomes impossible. Therefore the initial set of feature windows should consist of non-overlapping windows only.

4.2.3 Minor Eigenvalue Thresholding

A rich enough texture inside the window frames is necessary since these windows can be tracked easily and therefore contain motion information. Texture can be detected by various kinds of operators, like second-order derivatives or spatial frequency distribution. As mentioned in chapter 2 Tomasi & Kanade used a method based on eigenvalues of coefficient matrix $\mathbf{G}$.

Two small eigenvalues mean a roughly constant intensity profile within a window. A large and a small eigenvalue correspond to a unidirectional pattern. And two large eigenvalues can represent corners, salt-and-pepper textures, or any other pattern that can be tracked well. The fact now is that if both eigenvalues are sufficiently large, matrix $\mathbf{G}$ is well conditioned and above the image noise level, resulting in good trackable feature windows. So this condition is satisfied by easily thresholding the minor eigenvalue. The only problem is where to lie this threshold so that it holds in general.

The histograms in Fig.4.2 and Fig.4.3 represent the result of depicting integer rounded minor eigenvalues against their number of occurance. The axes are shortened to emphasize the main part. We thresholded the sorted minor eigenvalues with $\lambda_{th} = 10$, resulting in 199 features for the MILKPACK stream and 238 features for the CARD stream. The initial frames of the streams and the drawn windows are shown in

Fig.4.4 and Fig.4.5 . For comparison, Fig.4.6 gives image used in [Tomasi & Kanade, 1991a] with the computed histogram. Note the significant gap. We could not find any gap which seperates areas with high texture and areas with uniform background. Maybe this occurs with very specific image scenes, but we put some serious question-marks at this point.

Apparently the seperation between 'good' and 'bad' windows is not that easily to make. When simply thresholding the minor eigenvalue often a good set of trackable windows results. However, the danger still remains that also windows are selected which are in fact not trackable. A particular case is illustrated in Fig.4.7 through Fig. 4.9 . Fig.4.7 represents the starting image and Fig.4.8 and Fig.4.9 are respectively blow-ups of the initial frame and a frame 15 rotation steps later. As can be observed a window containing a straight line has been selected. However the tracking of the concerned window is incorrect. In some cases, although the algorithm is said to be designed to omit these kinds of features, eigenvalues of especially areas with a strong, roughly unidirectionally gradient lie too close to the 'good' windows to distinguish them by a single threshold.

4.2.4 Ratio Thresholding

For comparison we tried an additional selection method concentrating on the other eigenvalue as well. As stated the ratio between the two eigenvalues indicates specific areas. The procedure tests these areas (unidirectional pattern) against a predefined ratio. It rejects them from the initial set of feature windows (that is, after minor eigenvalue thresholding) when

$$\frac{\lambda_{max}}{\lambda_{min}} > \lambda_{maxratio}$$

In principle this means just an addition to the original method, making it less sensitive to different types of scenes since a ratio is concerned instead of an absolute value. Thresholding the minor eigenvalue needs less accuracy because the ratio thresholding rejects potential troublesome areas.

Note however that this is a *ad hoc* procedure and just meant to indicate that the creation of a solid set of feature points can be possible. A few extra experiments will yield a more generally applicable and founded method. In Fig.4.10 and Fig.4.11 , for the same two image streams MILKPACK and CARD, *both* eigenvalues of the original set of feature windows are depicted, sorted by the minor value. The graphs in Fig.4.12 and Fig.4.13 illustrate the deviation in ratio value. The high peaks indicate the possible unidirectional patterns. In Fig.4.14 and Fig.4.15 the result is shown after ratio thresholding, with $\lambda_{maxratio} = 10$. So what remains is *not* the most comprehensive set possible, but it avoids most of the potential problems.

4.3 Tracking

Analysis of tracking the feature windows can actually only be done after shape and motion recovery. Fig.4.16 and Fig.4.17 illustrate the tracking paths throughout 25 frames. The error in the CARD stream is caused by a window with an eigenvalue ratio of 9.44, just outside the inputted threshold value. The rotation of the objects can easily be verified. In Fig.4.18 and Fig.4.19 the tracked windows are drawn upon the last image of the series. Note that for image stream CARD the incorrectly tracked window follow the top edge of the card, ending up in the corner. During tracking, for image stream MILK four windows were rejected, reaching the maximum number of iterations (see section 4.3.3). For stream CARD ten windows were rejected.

In principle the selection method guarantees good trackable features, but it has a limited field of possible image scenes. Objects should contain enough texture and not too many protrusions, and situations like occlusion and reflections need special attention. These two particular cases are discussed in the next sections.

4.3.1 Occlusion

Although a small window size minimizes the occlusion problems, these phenomena will always occur, no matter what size. They are mainly caused by just disappearing and appearing of feature points due to camera/object motion. Especially an object with many protrusions or protuberances will give rise to occlusions because of the high chance of 'overshadowing' some feature points. So the software should detect and discard, or reconstruct, the occluding parts. As pointed out in chapter 2, in this stage we simply reject the occluded points from the measurement matrix $\mathbf{W}$.

In [Tomasi & Kanade, 1991b] the accumulated residu of the minimization procedure is thresholded in order to detect the occluding points. In [Tomasi & Kanade, 1992] occlusion is dealt with by filling in the incomplete measurement matrix $\mathbf{W}$. If a disappeared image point p in frame f is visible in at least three more frames f_1, f_2, f_3 and if there are at least three more points p_1, p_2, p_3 that are visible in the four frames f_1, f_2, f_3 and f then this point can be reconstructed. This can be done by first factor $\mathbf{W}$ to find partial motion and full shape solution and then propagate it to include motion for the remaining frame. Tomasi & Kanade are also preparing a special issue on this subject, as part of their paper series.

If the number of occlusions becomes too high, even for reconstruction, either the motion range or the initial frame are obviously not good. One should be careful, since not every image stream fits equally well in the proposed method. Our simplified camera motion (rotation only) is especially sensitive to occlusion. A multidirectional motion pattern is necessary in further experiments. Until that, a rough estimate by eye generally can avoid too many rejections.

4.3.2 Shiny Areas

As occlusion is impossible to avoid since we need motion to get 3D shape information, so are reflections caused by shiny areas. The reflections often result in a high gradient change, and so they are selected as good feature areas. However, most of the time they move into completely different directions than the camera or object motion. Fortunately, usually the reflections are straight line like areas, so most of them are rejected by ratio thresholding. Besides, the errors that occur can be directly percepted in three dimensions. Nevertheless, it is still better to avoid them all.

Solutions to this specific problem would mean an extra detector. Since distinguishing whether bright areas are caused by white or shiny surfaces asks for a *multiple* frame approach, the implementation probably would mean a significant slow down in the total algorithm.

4.3.3 Iterations

Iterations means here the number of interpolations needed to reach a threshold of accurateness. This number can serve as a primary indication of possible tracking errors. Typically the algorithm needs between five and ten iterations to achieve an accurateness of a thousandth of a pixel for the areas with a high minor eigenvalue. More iterations are needed for uniform areas and areas that get too heavely distorted due to camera rotation. These feature windows are rejected from the initial set since no reliable tracking is possible anymore.

4.4 3D Shape Recovery

Analysis of the actual 3D recovery performance is very limited since the recovery program had just been completed before starting typewriting this report. A rough estimate of its qualities can be made by observing the constructed 3D graph. In Fig.4.20 and Fig.4.21 the reconstructed 3D shape graphs of image streams MILKPACK and CARD are depicted. For the MILKPACK image stream a few guiding lines are drawn. The top view of image stream CARD allows us to verify the angle between the two sides. No doubt the shape has been recovered well, however, measurements need to be done to get an impression of the accuracy.

5. Conclusions

Though not a definite opinion can be formed simply because the analysis is not ample enough, a few comments will be listed here.

- Selection of possible feature points based on eigenvalue calculation gives reliable results in most cases on the condition that an appropriate threshold can be found. Nevertheless eigenvalues of *all* possible windows need to be computed. When done straightforward, this part consumes a considerable amount of computation time. Sophisticated methods which for instance skip the background areas, should be applied to save time.

- Finding a general threshold value for easy selection seems not possible by only accounting minor eigenvalues and therefore needs additional help. At first sight, the involvement of the other eigenvalue seems to cover most of the difficulties, making the selection of features more reliable.

- Tracking of the selected feature points is easy, though might come in problem when the initial set contains false feature points or occlusion occurs during the tracking.

- Adequate handling of the main problem of occlusion needs definitely implementation. The phenomenae occurs frequently and results easily in unreliable shape and motion recovery. The recently published article [Tomasi & Kanade, 1992] proposes a plausible propagation method to estimate the disappeared feature points.

- 3D shape and motion recovery using the factorization method is fast and seems reliable at first sight. The claimed excellent performance under noisy conditions should be tested.

- The major drawback of the method is that a whole sequence is processed at once, and not whenever new images become available. Future research might help to overcome this problem. An advantage however, is the fact that the method does not assumes *a priori* information about either shape or motion, which makes it more generally applicable than similar recovery methods.

Appendix

Here, a short outline of the input and output files, the written programs produce, is given. The word **stream** indicates the name of the image stream, and is given as an argument to the program call. A sequence of grabbed images **stream.xxx** should be in the specified directory. Examples :

```
LambdaCalculation box
FeatureSelection /buildings/house      if in other directory.
```

Be careful with the diskspace when using **WindowDrawing**, since this program produces memory consuming image files.

Program Name	Input File(s)	Contents
LambdaCalculation	stream.sss	Visilog3 Image File (sss = Start Index)
FeatureSelection	stream.lambda	Sorted Eigenvalues
FeatureTracking	stream.fps.sss stream.xxx	Feature Windows of Frame sss Visilog3 Image File (xxx = Index Num.)
WindowDrawing	stream.fps.xxx	Feature Windows of Frame xxx
ShapeRecovery	stream.data	Feature Windows of All Frames

Program Name	Output File(s)	Contents
LambdaCalculation	stream.lambda stream.hist	Sorted Eigenvalues Counted Eigenvalues for Histogram
FeatureSelection	stream.fps.sss stream.minmax	Feature Windows of Frame sss Sorted Eigenvalues for Histogram
FeatureTracking	stream.fps.xxx stream.data	Feature Windows of Frame xxx (xxx = Index Number) Feature Windows of All Frames
WindowDrawing	stream.FPs.xxx	Feature Windows on File stream.xxx
ShapeRecovery	stream.shape stream.motion	3D Shape Information 3D Motion Information

References

[Tomasi & Kanade, 1991a]

TOMASI, C., AND KANADE, T., Shape and motion from image streams : a factorization method - 2. Point features in 3D motion. Technical report CMU-CS-91-105, Carnegie Mellon University, Pittsburg, PA, January 1991

[Tomasi & Kanade, 1991b]

TOMASI, C., AND KANADE, T., Shape and motion from image streams : a factorization method - 3. Detection and tracking of point features. Technical report CMU-CS-91-132, Carnegie Mellon University, Pittsburg, PA, April 1991

[Tomasi & Kanade, 1992]

TOMASI, C., AND KANADE, T., Shape and motion from image streams under othography : a factorization method.,*International journal of computer vision*, vol. 9, num. 2, pp. 137-154, 1992

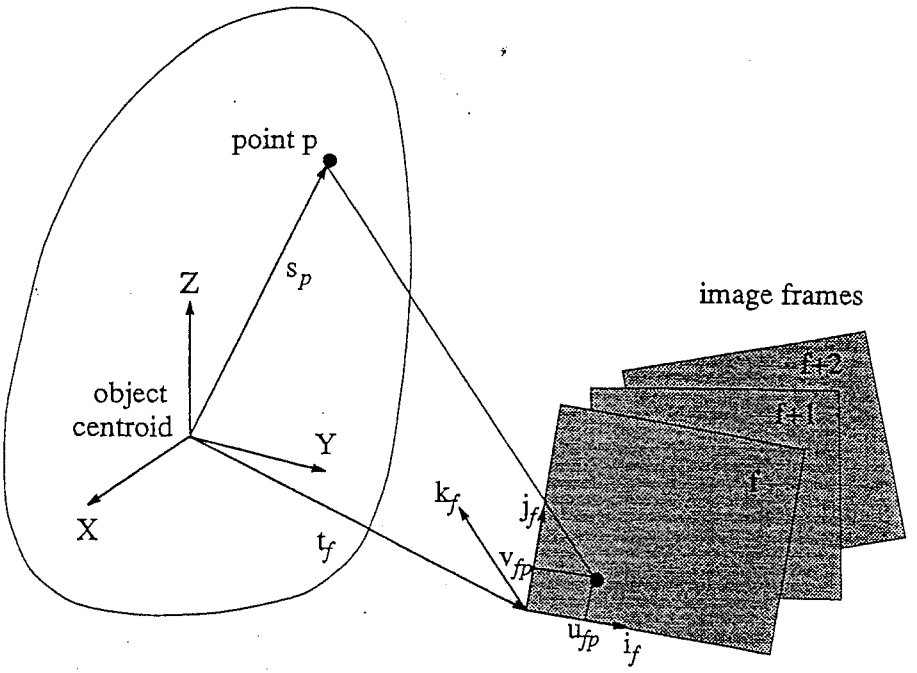

Fig. 2.1 The two reference systems [Tomasi & Kanade, 1992].

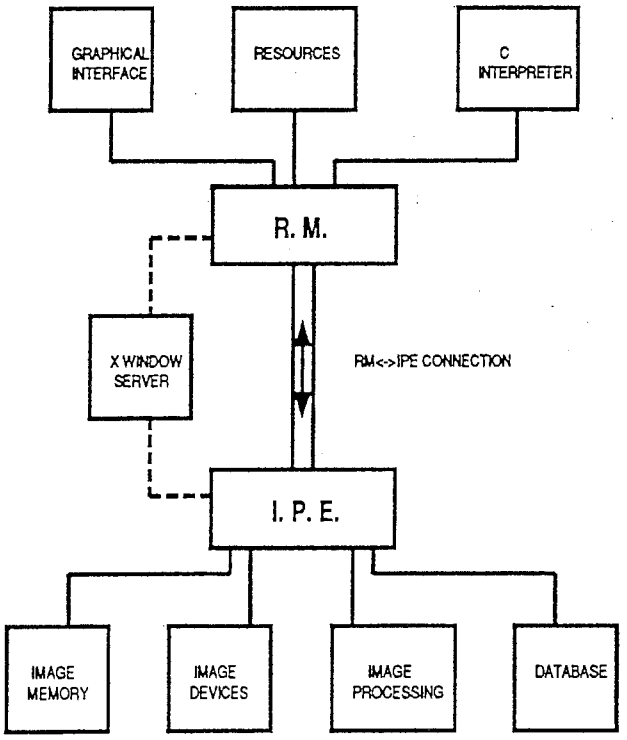

Fig. 3.1 The organisation of Visilog.

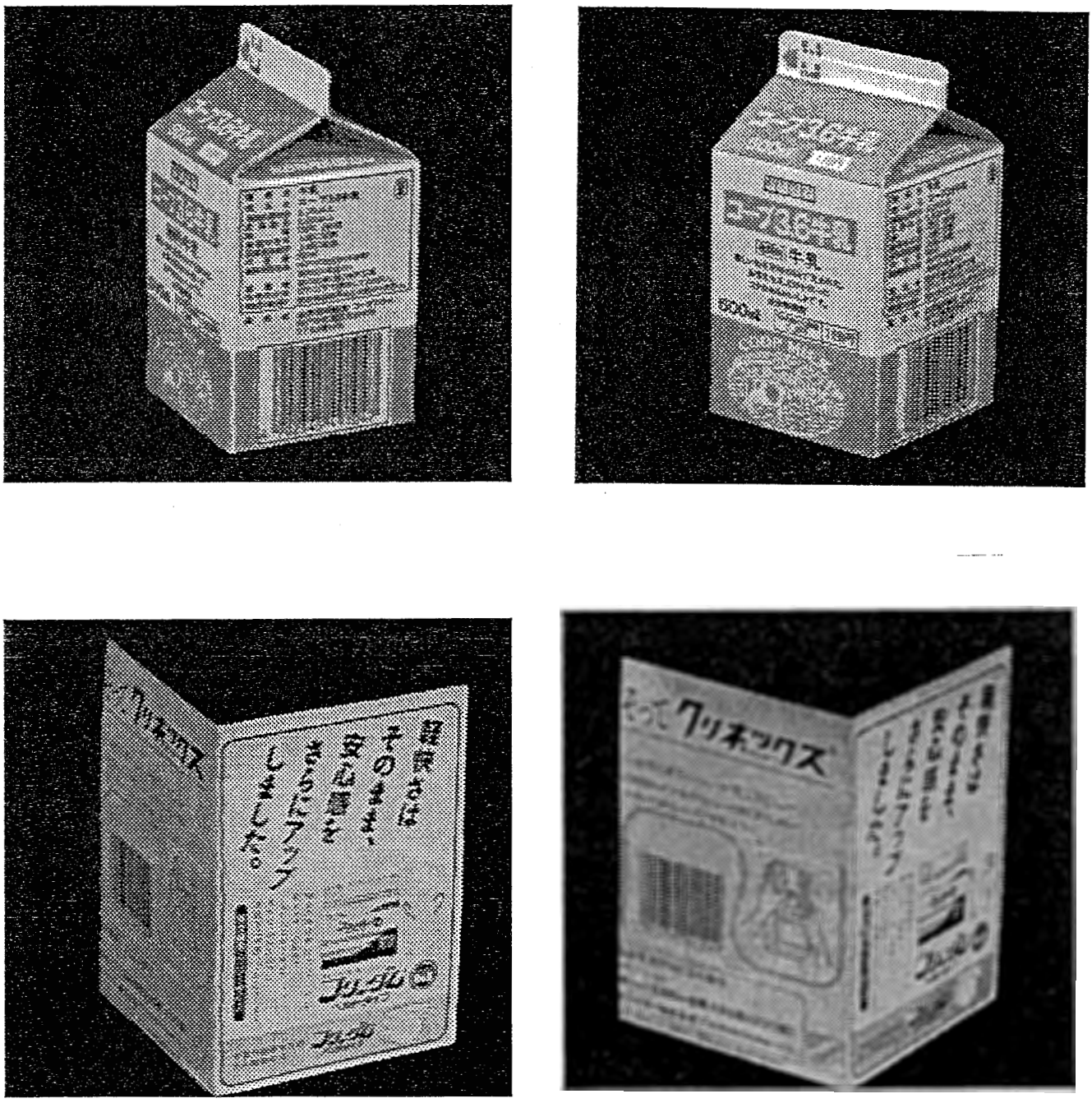

Fig. 4.1 Start and ending frames of image streams : MILKPACK and CARD.

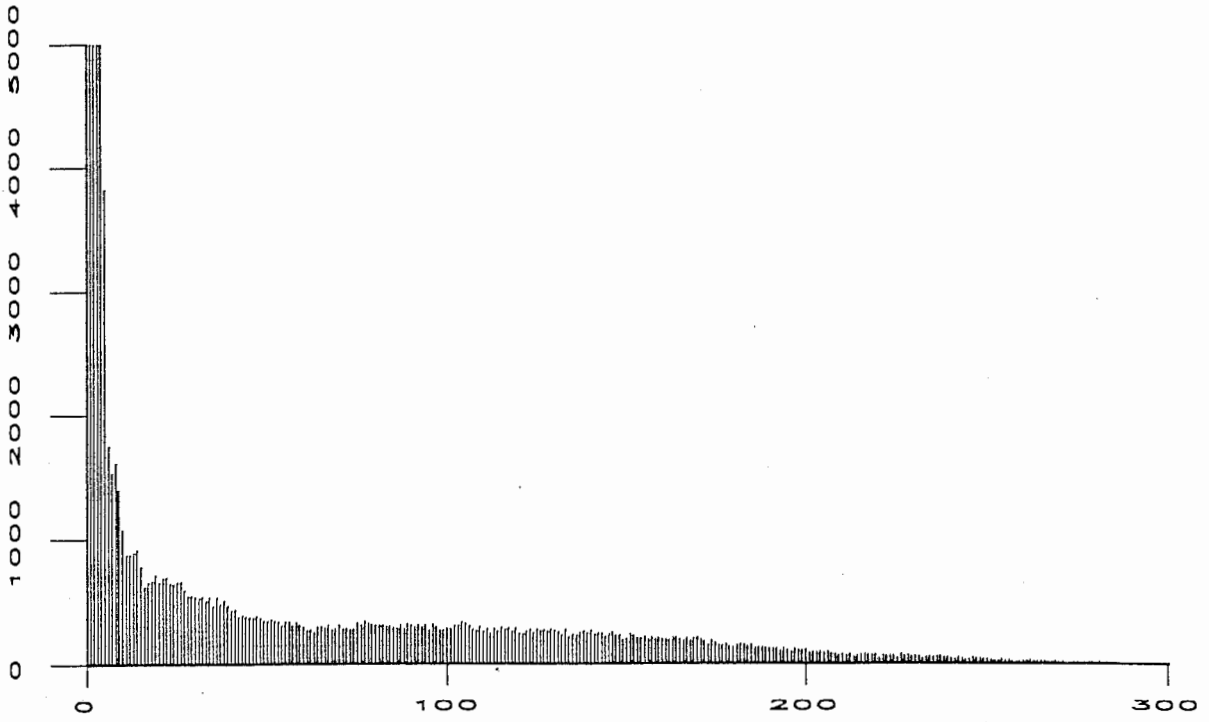

Fig. 4.2 Counted minor eigenvalues against their number of occurance for image stream MILKPACK.

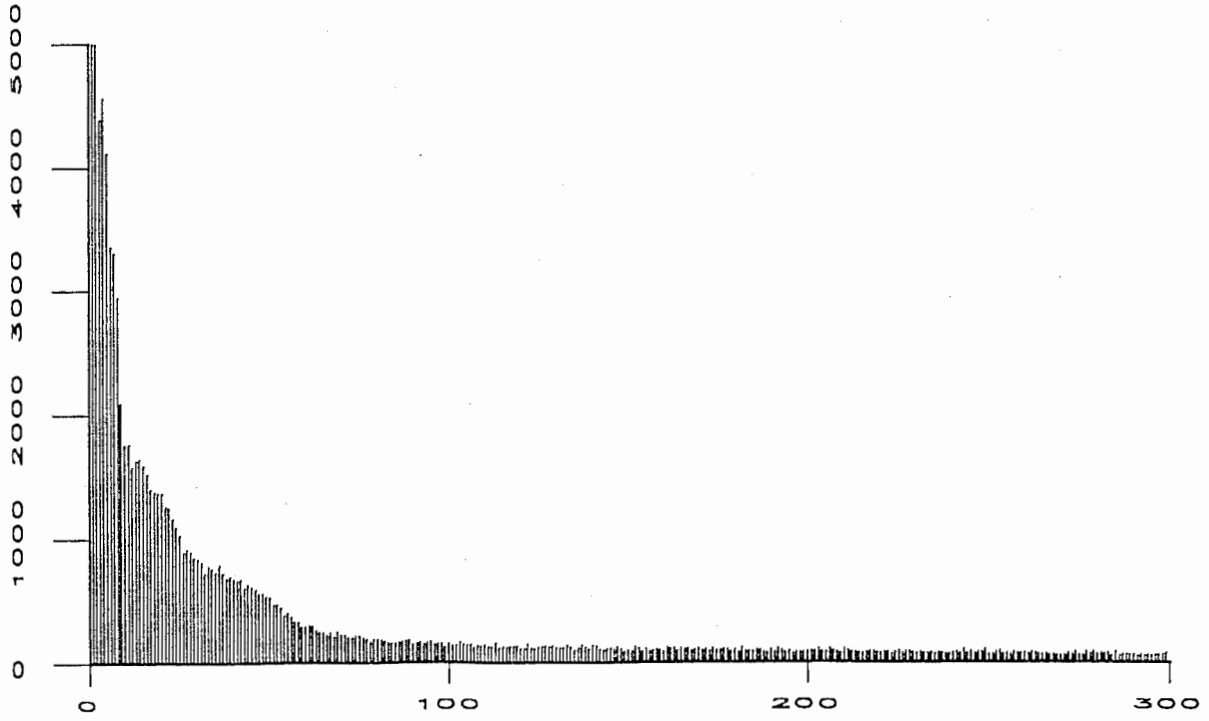

Fig. 4.3 Counted minor eigenvalues against their number of occurance for image stream CARD.

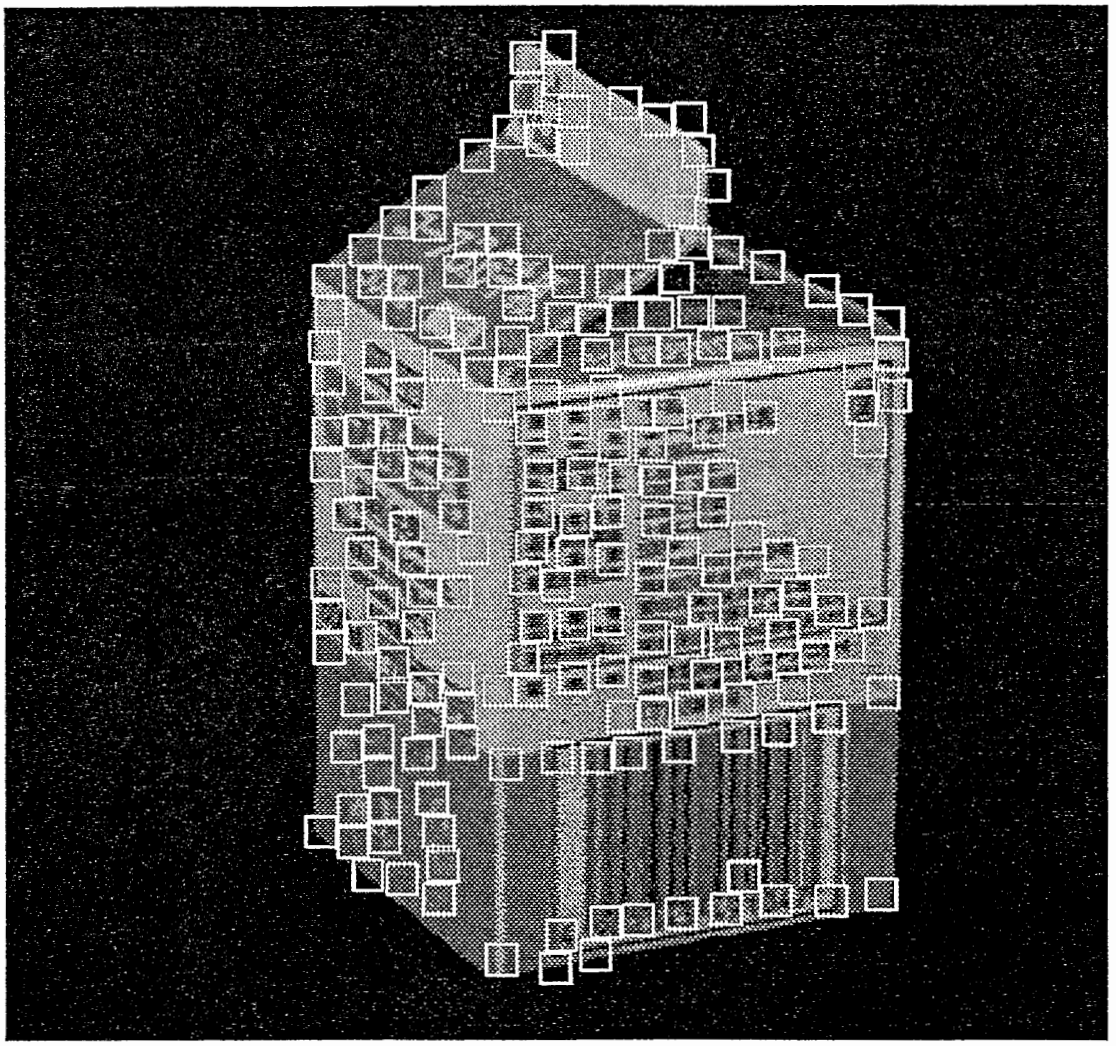

Fig. 4.4 Result of thresholding minor eigenvalues of image MILKPACK ($\lambda_{th} = 10$).

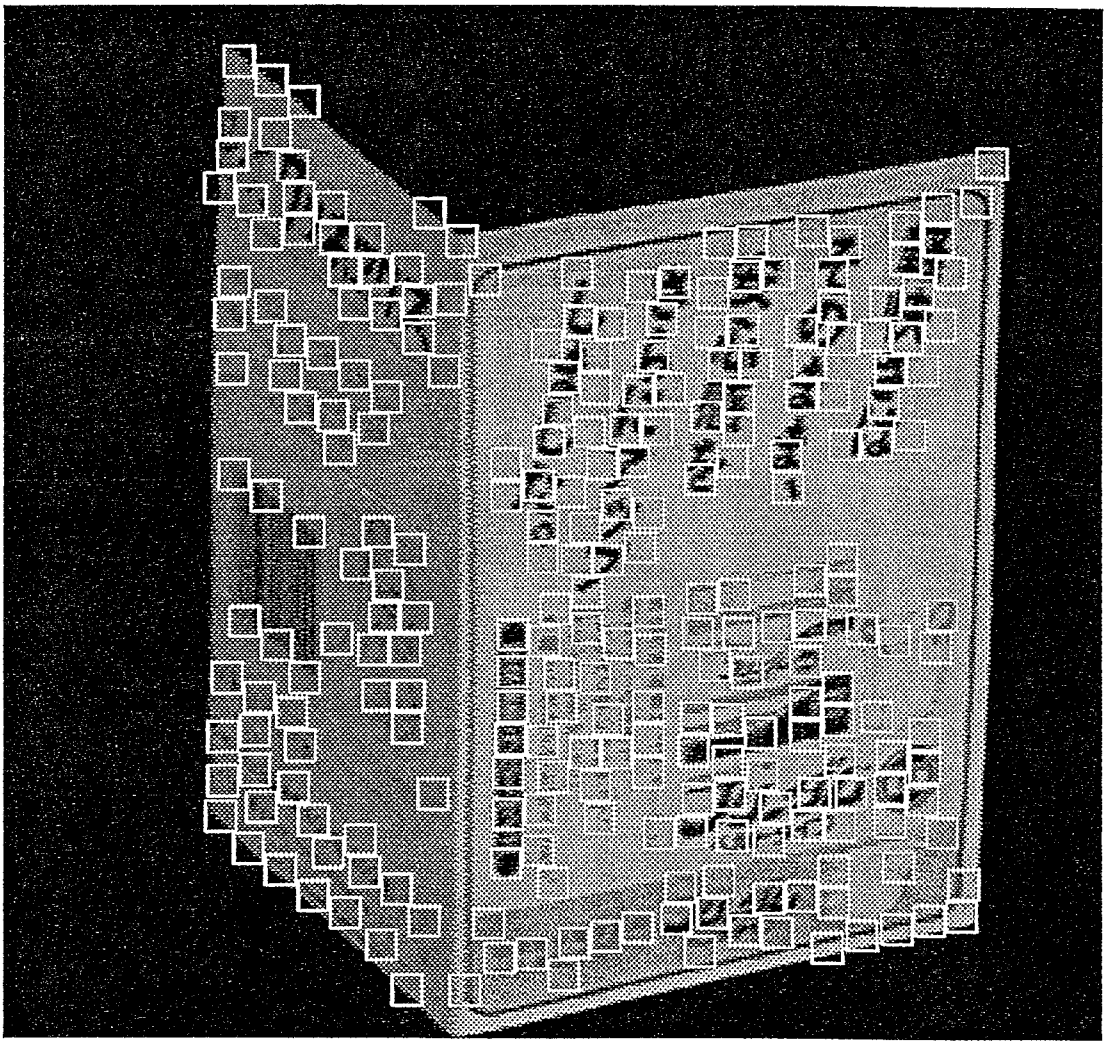

Fig. 4.5 Result of thresholding minor eigenvalues of image CARD ($\lambda_{th} = 10$).

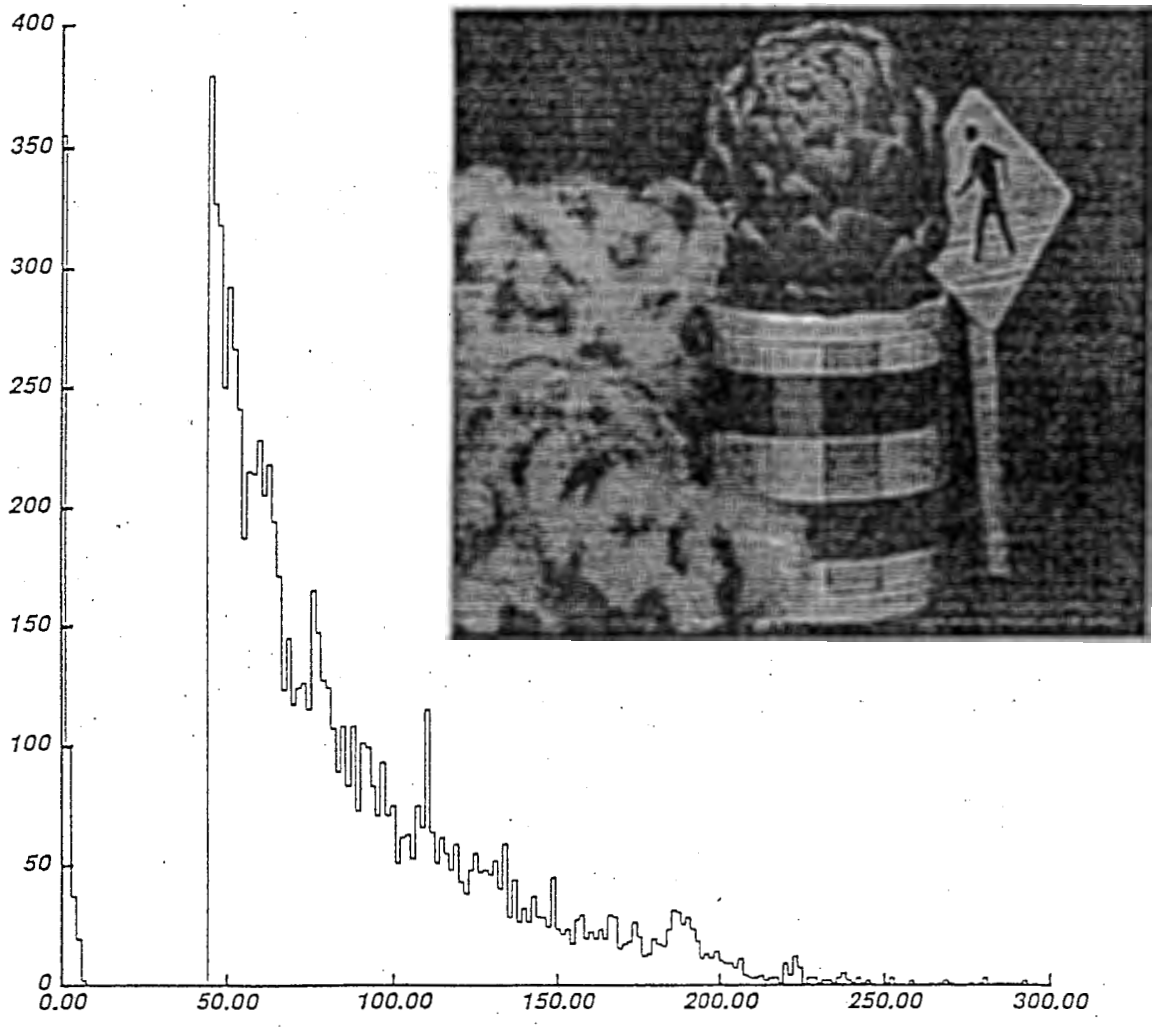

Fig. 4.6 Image and histogram from [Tomasi & Kanade, 1991b].

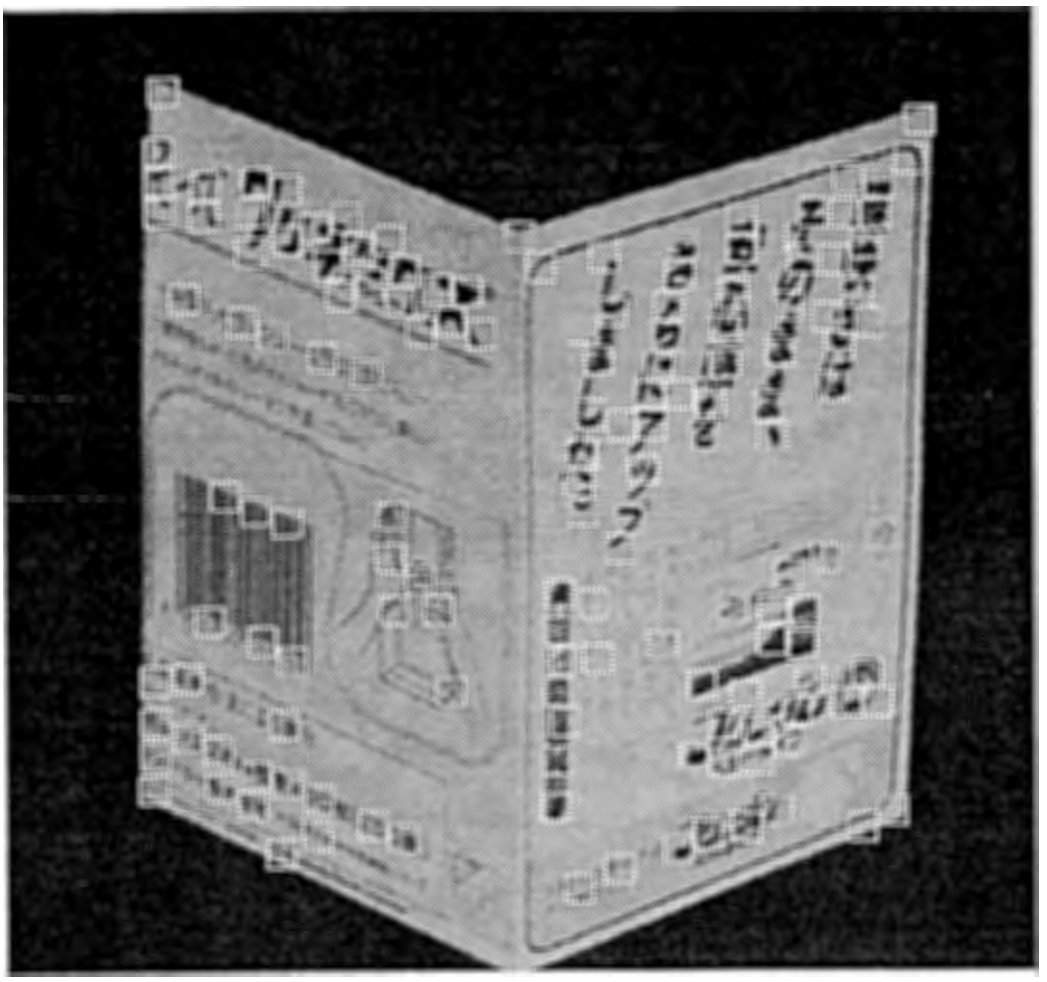

Fig. 4.7 Image stream : CARD.

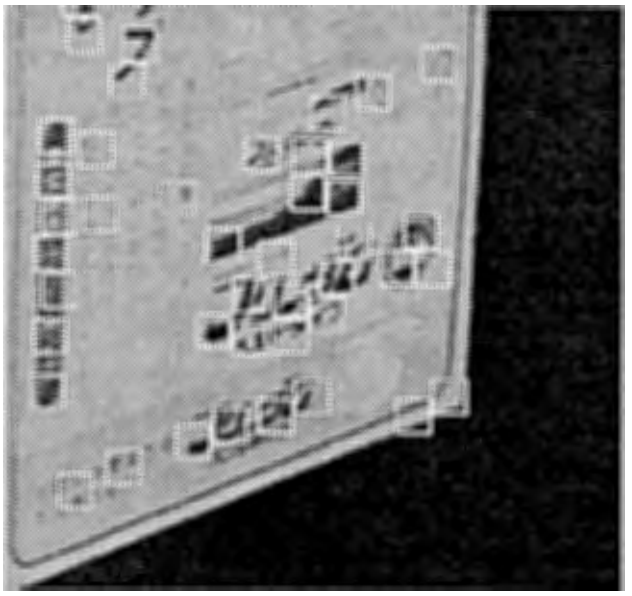

Fig. 4.8 Blow-up : Frame 1.

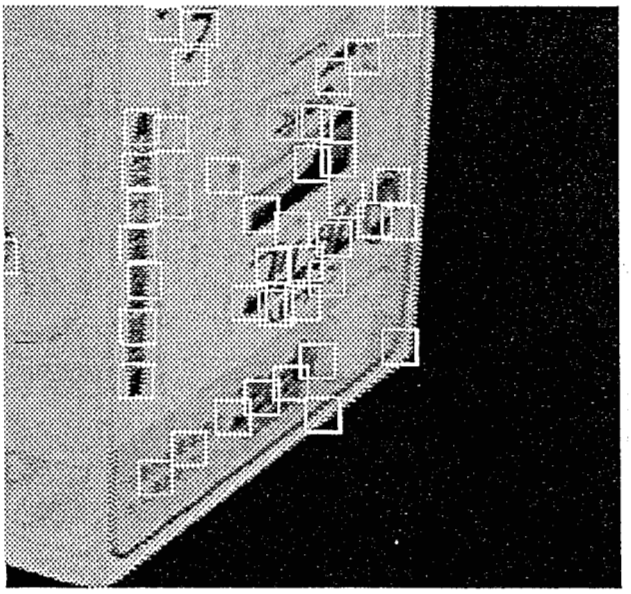

Fig. 4.9 Blow-up : Frame 15.

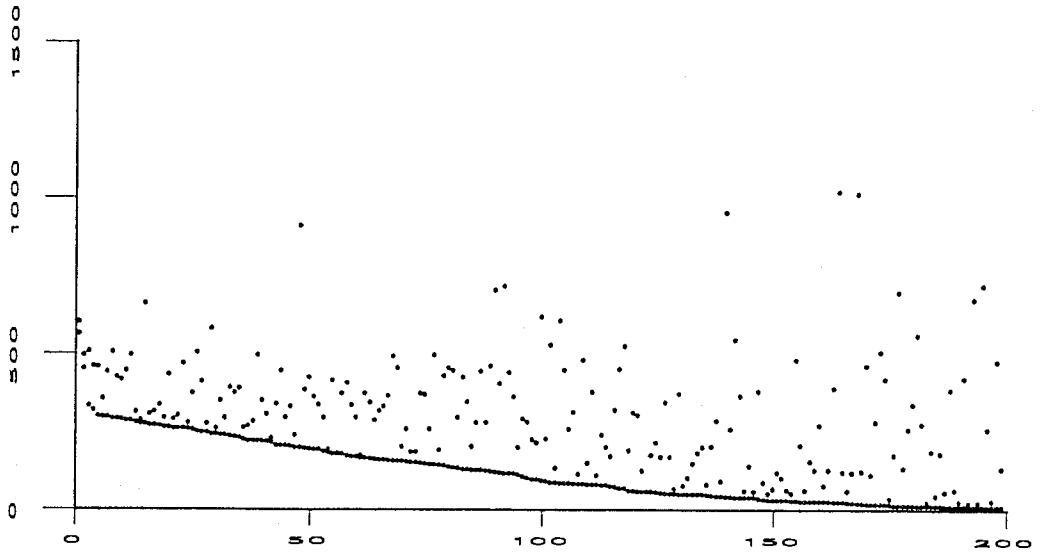

Fig. 4.10 First 200 sorted eigenvalues of image stream MILKPACK.

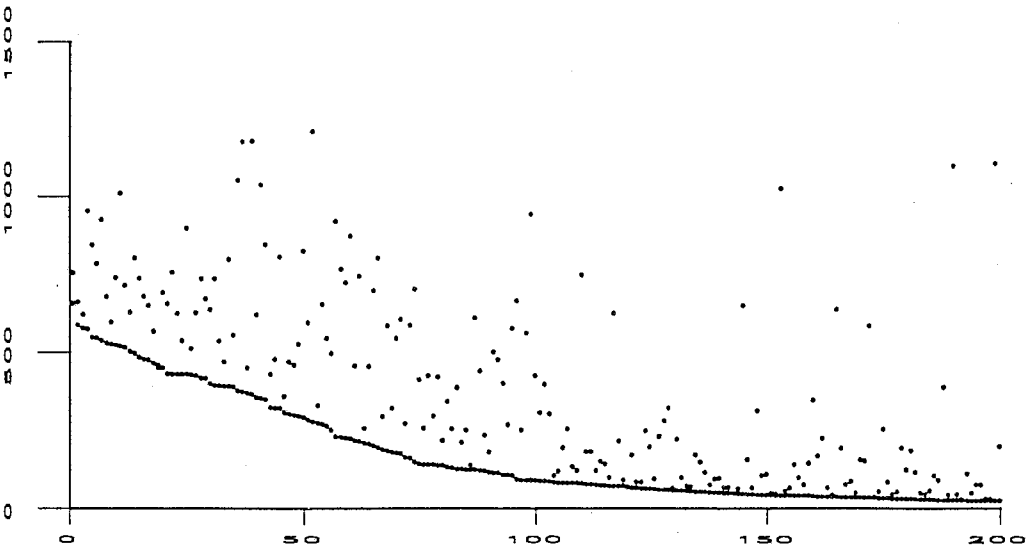

Fig. 4.11 First 200 sorted eigenvalues of image stream CARD.

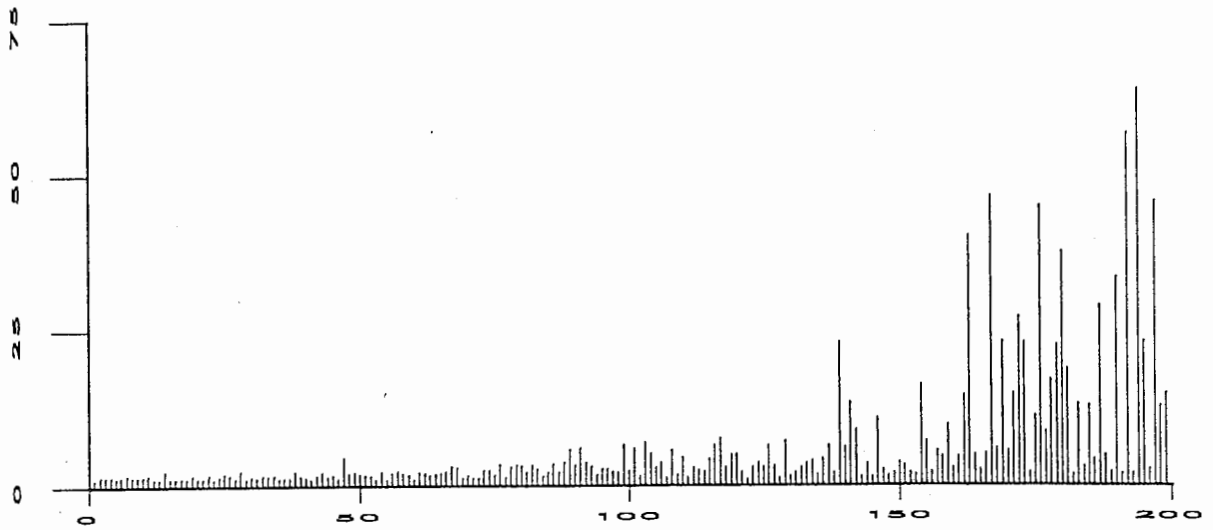

Fig. 4.12 First 200 ratio values between max. and min. eigenvalues for image stream MILKPACK.

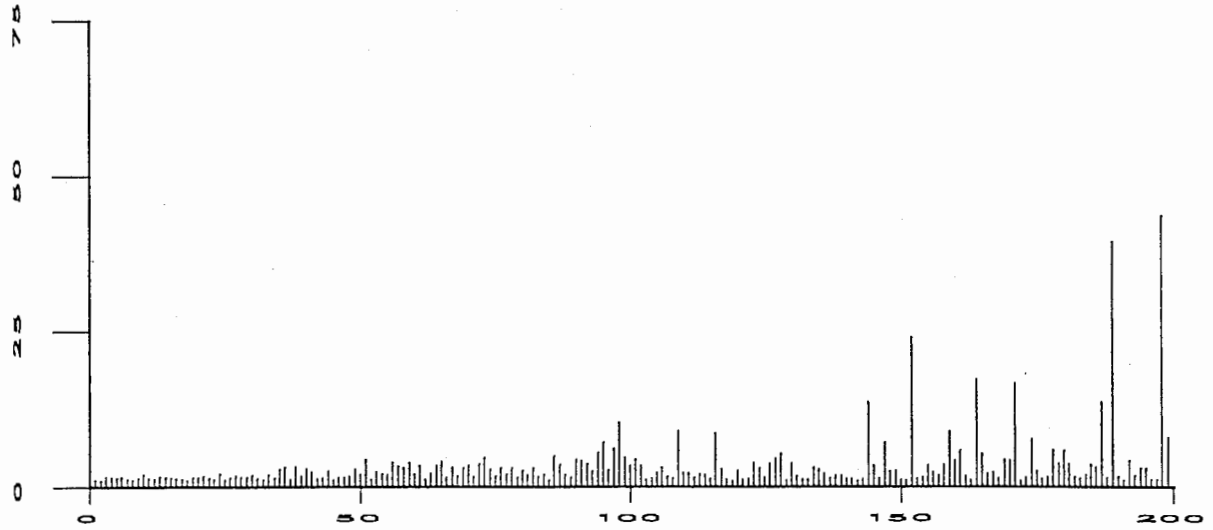

Fig. 4.13 First 200 ratio values between max. and min. eigenvalues for image stream CARD.

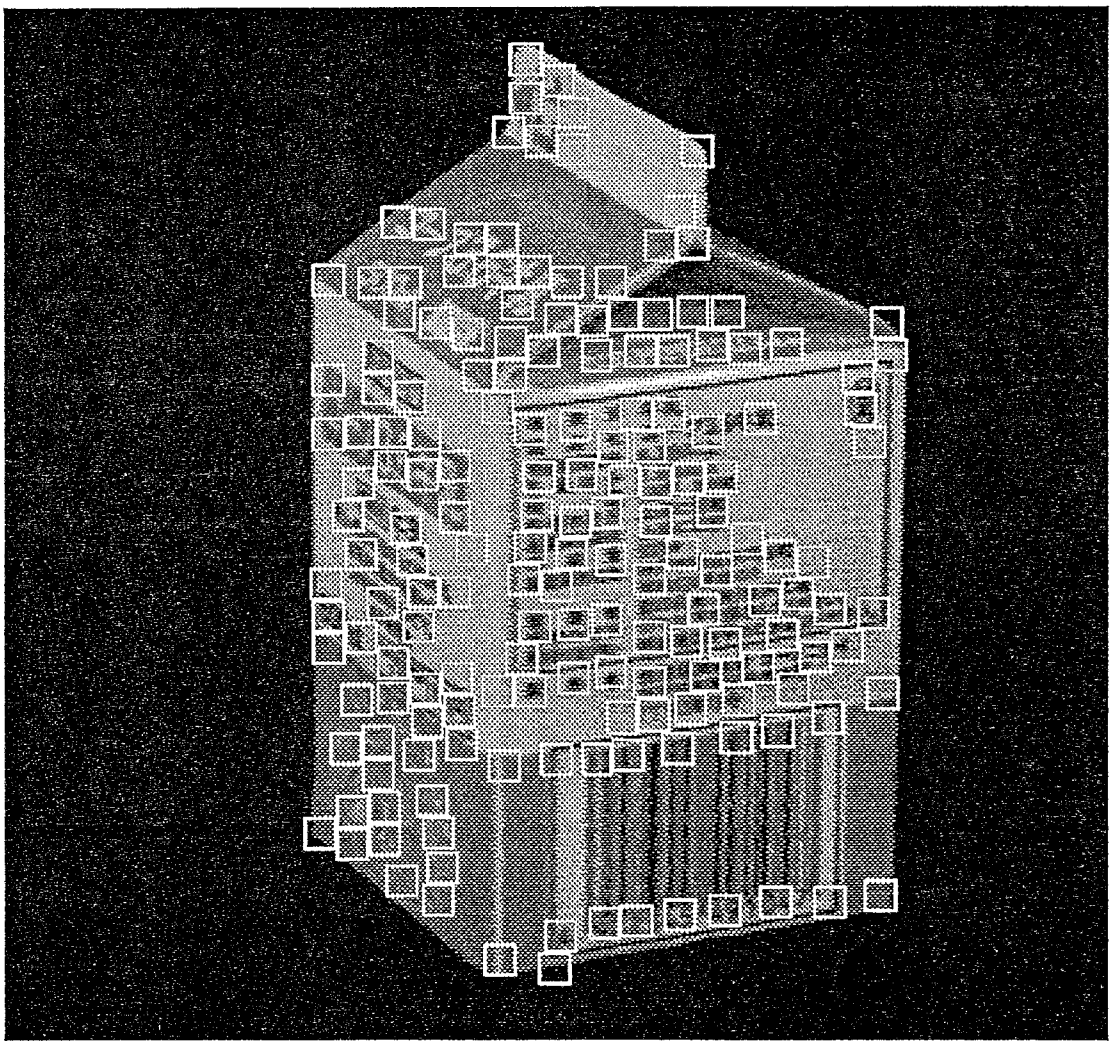

Fig. 4.14 Remaining windows (173) after *ratio thresholding* for image stream MILKPACK. With $\lambda_{maxratio} = 10$.

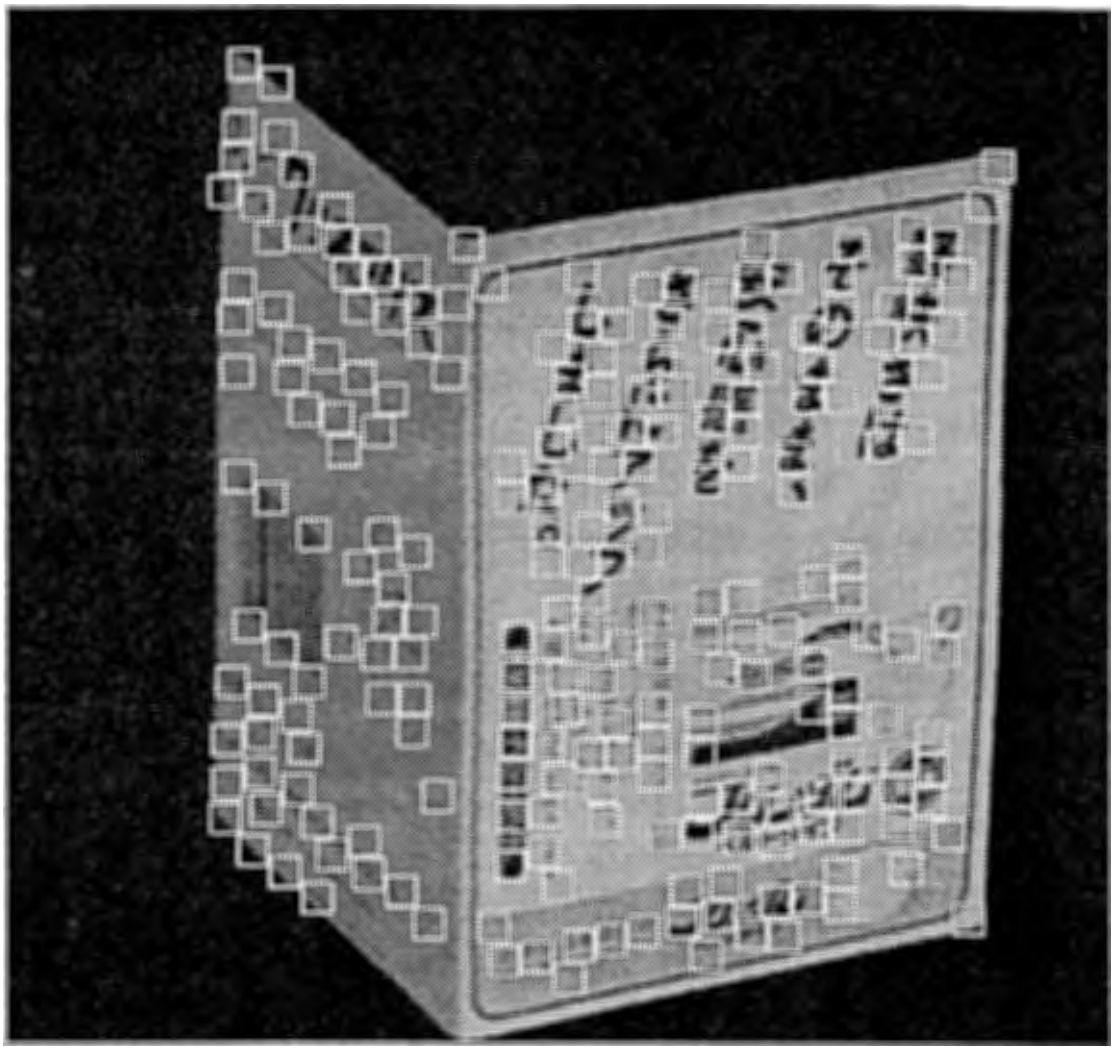

Fig. 4.15 Remaining windows (238) after *ratio thresholding* for image stream CARD. With $\lambda_{maxratio} = 10$.

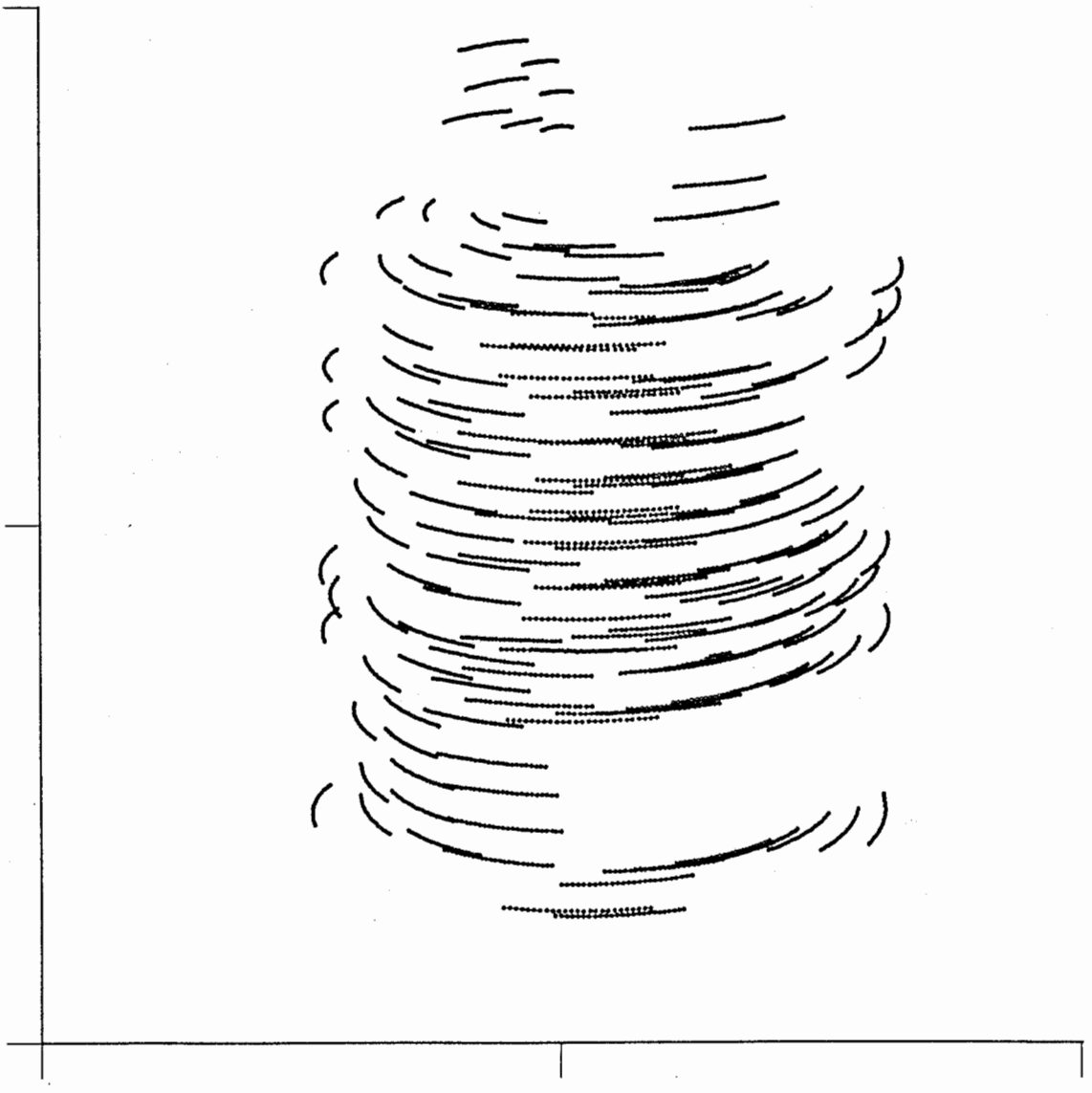

Fig. 4.16 Tracking results for image stream MILKPACK.

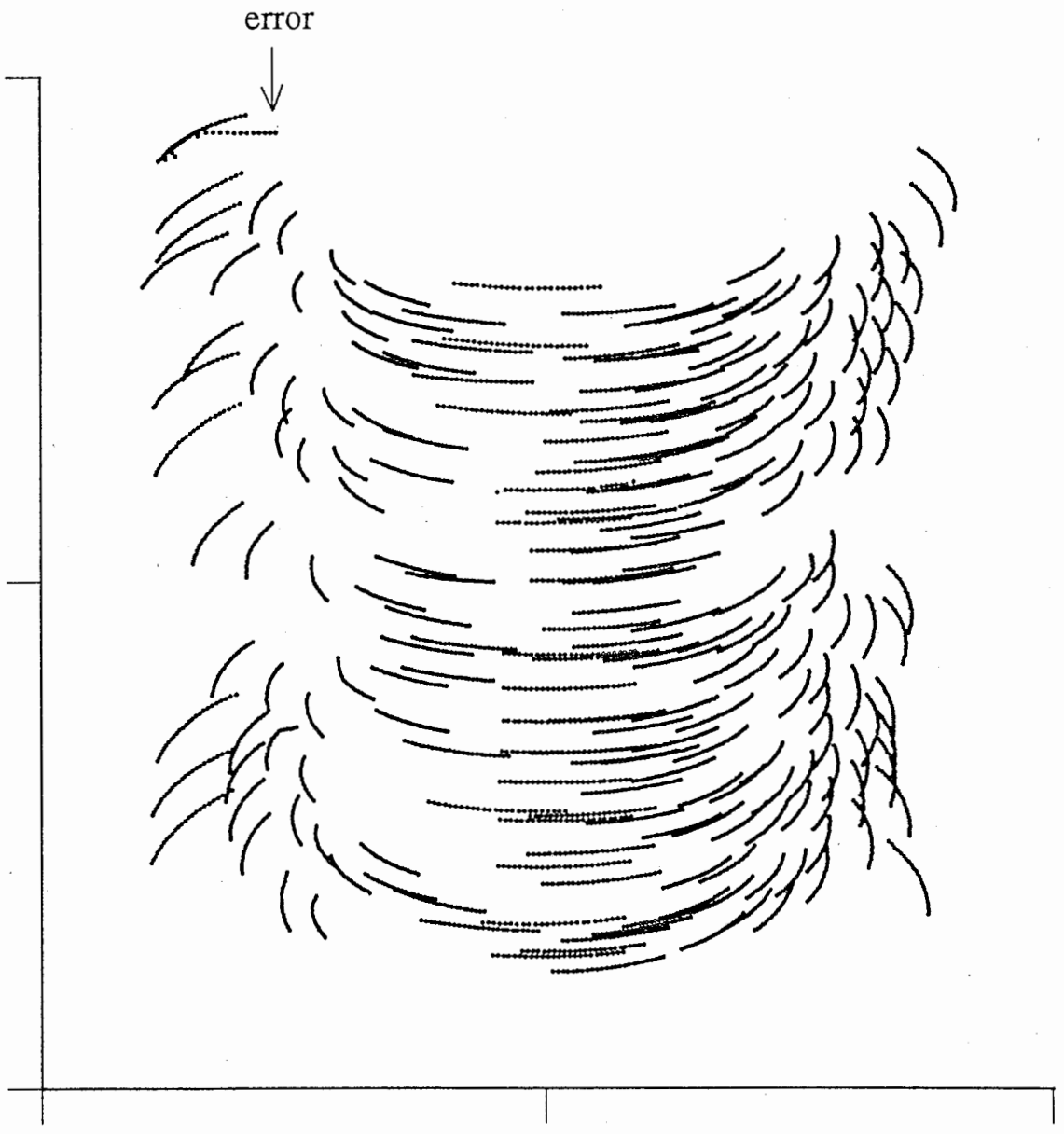

Fig. 4.17 Tracking results for image stream CARD.

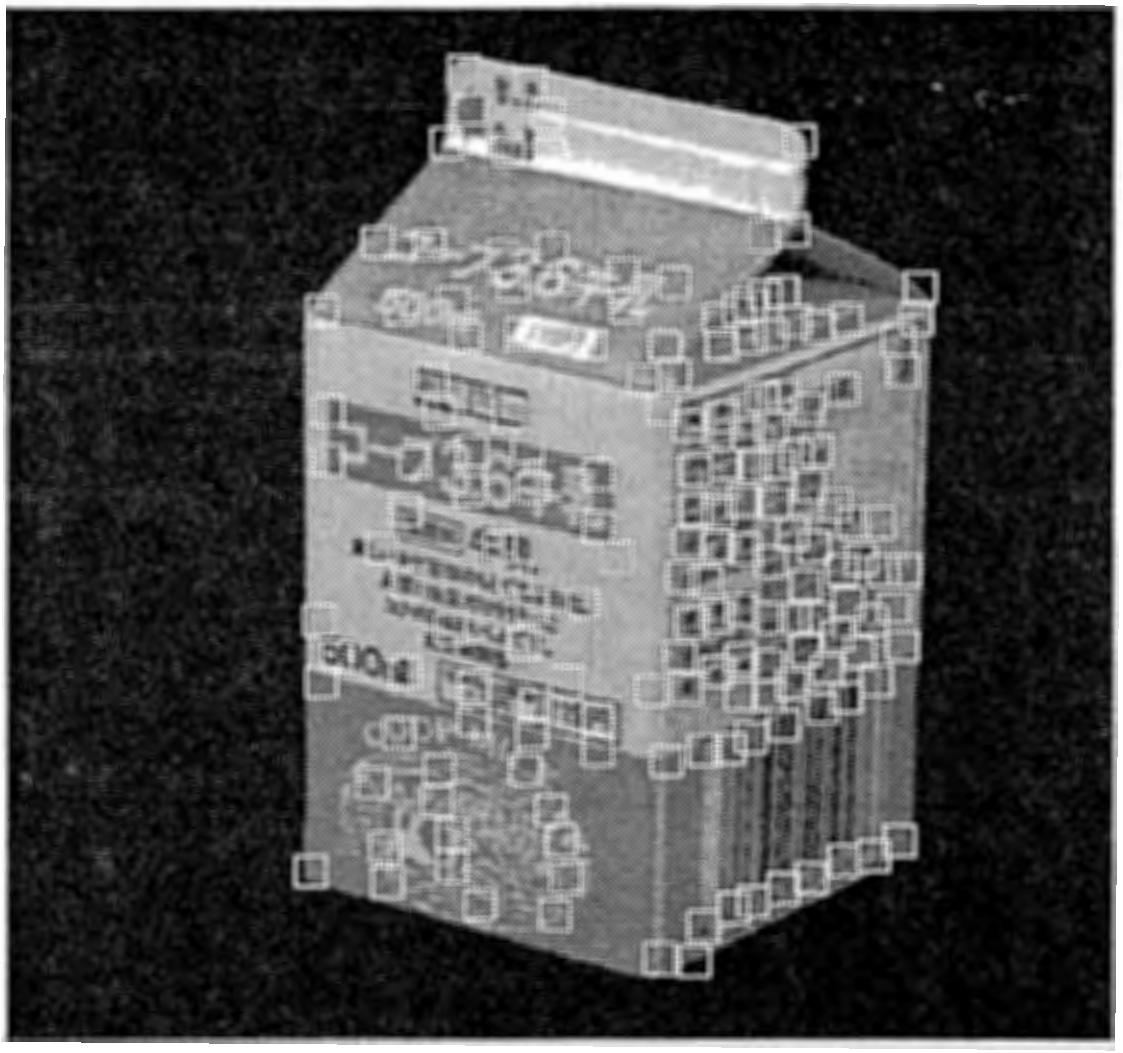

Fig. 4.18 Final frame of image stream MILKPACK : 169 windows remaining.

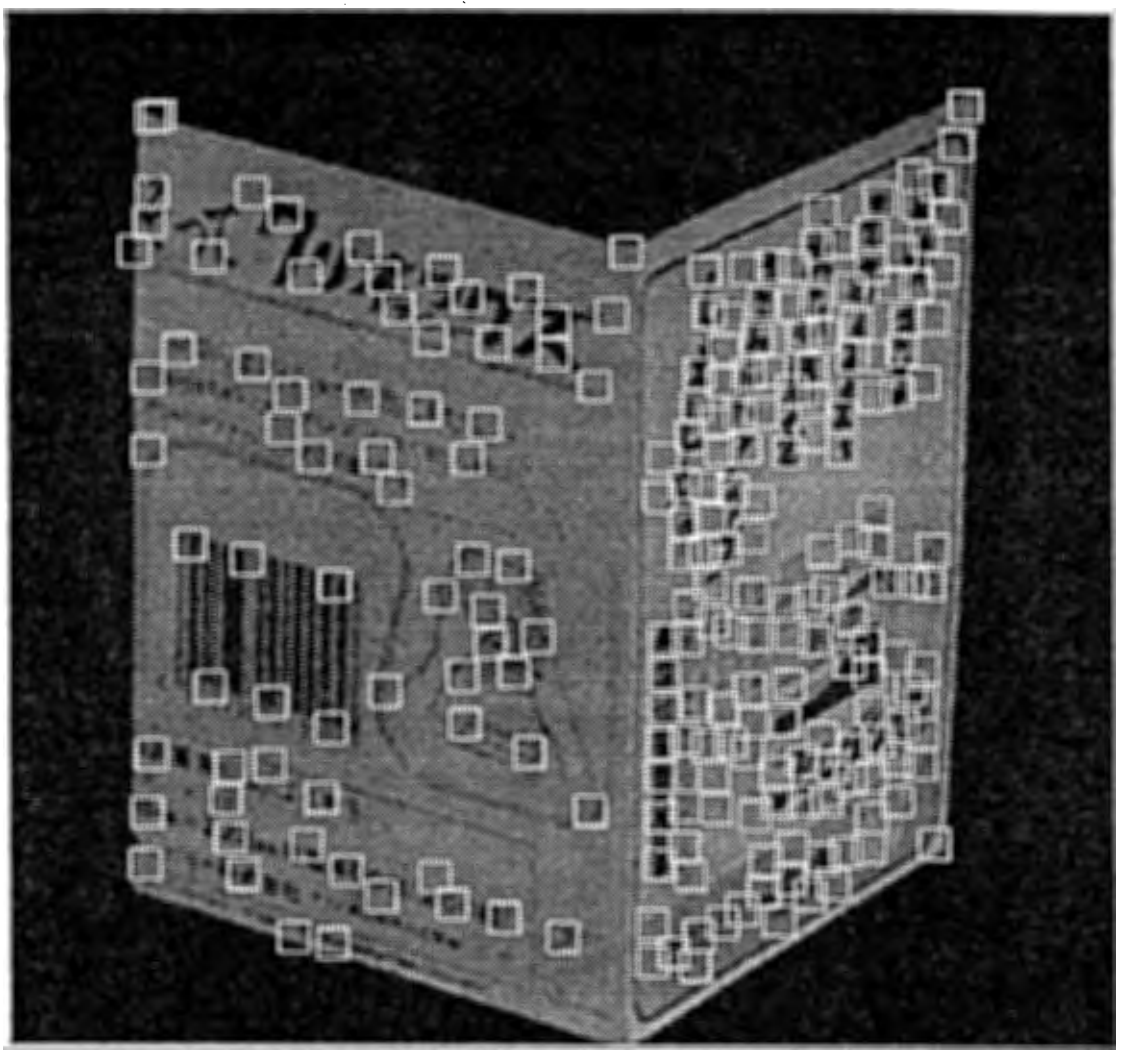

Fig. 4.19 Final frame of image stream CARD : 228 windows remaining.

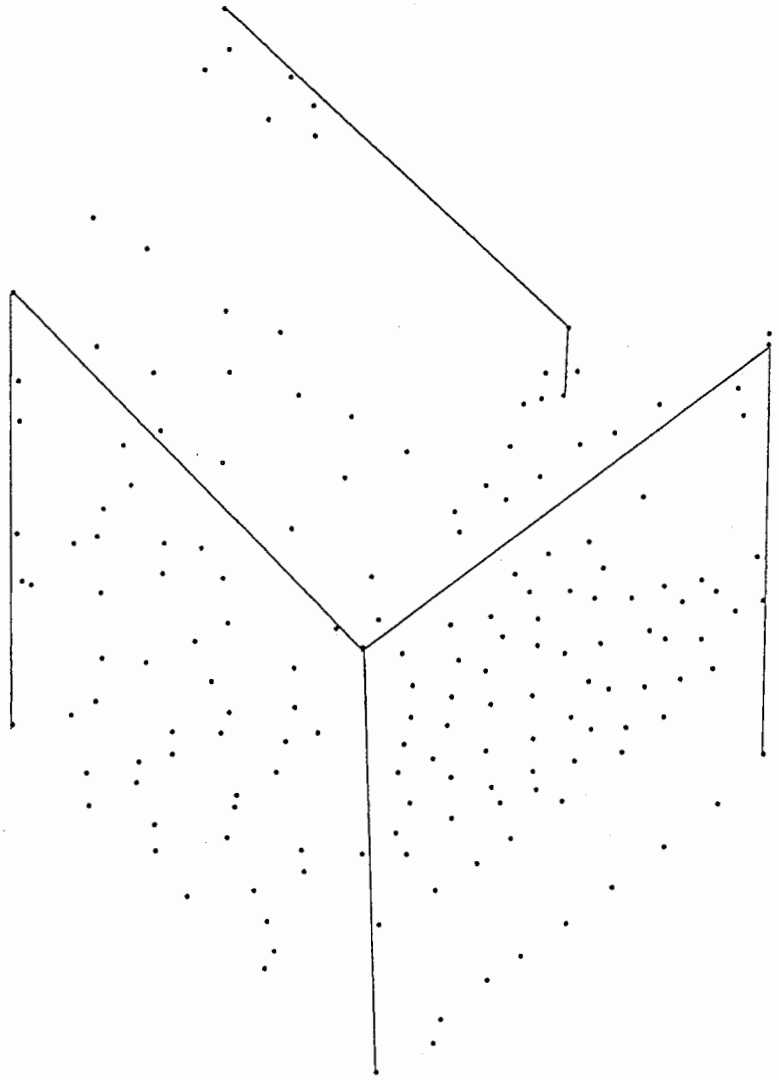

Fig. 4.20 Recovered 3D graph of image stream MILKPACK.

Fig. 4.21 Recovered 3D graph of image stream CARD.