MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence Memo. 136
PDP-6 Program Memo. July 1967.

Matrix Inversion in LISP

John L. White

Very shortly there will appear on the vision library tape a file named @IAS which is a collection of compiled SUBR's for performing general matrix row reduction and inversion. For an array A, a call

(IAS A NEW N M)

performs gaussian row reduction on the first N rows of the array A (and in fact operates on only the first M columns); so that if M>N then the N+1 st through the Mth columns of the output array contain the solutions to the implicit M-N+1 systems of NxN simultaneous linear equations, while the first N columns contain the inverse matrix of $\begin{bmatrix} A_{11} & \cdots \\ \vdots & A_{NN} \end{bmatrix}$. If NEW is 'T' then a new array of size NxM is declared and the answers are stored in it leaving the input array A alone; if NEW is 'NIL' then the output array is stored directly over the input array and no new array declarations are done.

Currently, maximization of pivotal elements is not done; thus IAS will give wrong answers on certain numerically ill-conditioned matrices even though they be non-singular. It is possible to remedy this problem, at some expense, if necessary. IAS also uses a portion of binary program space for temporary storage and may give an error message if not enough space is available.

A visual scene is composed by the optical image of a group of bodies in a determinate position. Nevertheless, when such a scene is "seen" by a computer through a film spot scanner, vidissector, or similar device, it will look more like a two-dimensional array of numbers. As such, it could be described by a function of two variables.

At a higher level, a scene could be meaningfully described as a conglomerate of points, lines and surfaces, with properties (coordinates, slopes, ...) attached to them.

Still a more sophisticated description could be done in terms of the bodies or objects which compose such scene, indicating their positions, inter-relations, etc.

A program is described here which takes as its input a collection of lines, vertices and surfaces describing a scene presumably composed by three-dimensional objects, and identifies the bodies which form such a scene. In other words, a scene (in a somewhat abstract representation) is analyzed and decomposed into the 3-dimensional objects which form it.

The program does not need to know the form (the model, the pattern) of the objects which are likely to appear: the scene is not searched for cubes, wedges or houses, with an a-priori knowledge of the form of these objects; rather, the program pays attention to configurations of surfaces and lines which would make "plausible" 3-dimensional solids, and in this way 'bodies' are identified.

The program is restricted to scenes formed by straight lines (no curved surfaces allowed), where no shadows or noise are present. It has been tested in rather complicated scenes composed by rather simple objects. Examples are given. Format of the input and output are described.