MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working paper 127 , July 1976

Hand Eye Coordination

by

Glen Speckert

This paper describes a simple method of converting visual coordinates to arm coordinates which does not require knowledge of the position of the camera(s). Comparisons are made to other methods and two camera, three dimensional extensions are discussed. The single camera method for converting points on a tabletop is used by Marc Raibert and Glen Speckert in a working hand-eye system which recognizes objects and picks them up under visual guidance. This was implemented on the MIT Micro-Automation PDP 11/45 using a low speed vidicon and a Scheinman arm.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75C-0643-0005.

Almost any arm manipulation task can be made easier or more useful if accompanied by a visual component. Often however, the software of a system involving arm manipulation is such that its author knows little about visual systems. Similarly, researchers in vision tend not to venture into the complex field of robotics. Ultimately (hopefully), these fields will both be on a high enough level that a complex visual-mechanical interdependent system can easily be produced. Even then the visual system must interact with the mechanical one in a straightforward way, and one which will minimize errors, rather than propogate them. Some possible sources of error from the visual system include spherical distortion due to the lens, X-Y distortion due to the non-square pixels, random distortions due to lens flaws and irregularities, and smearing and loss of resolution inherent in the device itself. Problems encountered in the arm system include errors in converting from physical coordinates to joint angles, actual errors in joint placement resulting from mechanical problems, noise in the signals, and many others. Hence what is needed is some magic convertion package which will overcome all these difficulties, and enable the visual system to interact easily and accurately with the arm system.

There are a few basic approaches to this problem. One is similar to the way that the arm alone is able to "know" where it is, namely to rigidly attach the visual system to the arm system, or attach both to the same table. This usually implies finding the angle that the camera is at,

and its rotation, then calculating the linear transformation. Another basic method is to provide relative feedback, i.e. visual information of arm position relative to the object to be picked up. This implies being able to accurately and quickly find the location of both the arm and the object, probably from more than one camera.

Having only one vidicon, the only way to be able to convert from vision space to arm space is to assume the points lie on a table, or are in some other way constrained in one dimension. In A. I. Working Paper 34 (December 1972), Berthold Horn has described the matrix mathematics necessary to convert from visual space to arm space given that the angle of the camera to the object plane (table) and the rotation of the camera are known. Unfortunately the only practical method of obtaining these quantities is to look at some points on the table and compute the angle and rotation from the visual image of the points. This leads to a transformation that will only work if the visual field is uniform relative to the points used to compute this angle and rotation, and if the arm can accurately go to real coordinates x,y. (Real space is the same as arm space except for the errors due to the arm itself). We have already stated that the visual field is not uniform, and neither is arm space. Thus in spite of all the complex mathematics (or maybe because of it), this method may not be accurate enough to actually pick up the object viewed.

The method described below only assumes that the visual field is very locally uniform. Just how uniform one assumes is inversely

porportional to the number of points one wants to incorporate into the calibration table. Basically the method used requires placing the ball at several calibration points (9 to 16 points were used) and using a subset of these calibration points that is visually near the visual image of the point to be converted. This has the advantages that the camera position need not be known, the visual distortions are less likely to cause failure, and any consistant errors in arm placement is automatically taken into account.

The vidicon is placed where it can scan the entire working area of the arm. Next, the arm places the ball in key positions, and a calibration table of arm and visual coordinates is generated. The vidicon must then remain in this orientation relative to the arm, and thus this is a modified "rigidly attached" method. See figure 1 for a diagram of the setup.

The arm has the ball initially, and sets it in a known location (at known arm coordinates). The arm then gets out of the way so that the vidicon can locate the ball, and the visual and arm coordinates are noted in a table. The arm then picks up the ball (assuming it has not rolled away), and moves it to another location where its visual and arm coordinates are again noted. In this way, the irregularities in the lens and the errors in the arm are all incorproated into the calibration table. Any number of points can be put into this table, and the more calibrated points there are, the weaker the assumption about locally uniform visual field need be. In practice, nine to sixteen points were used. See figure

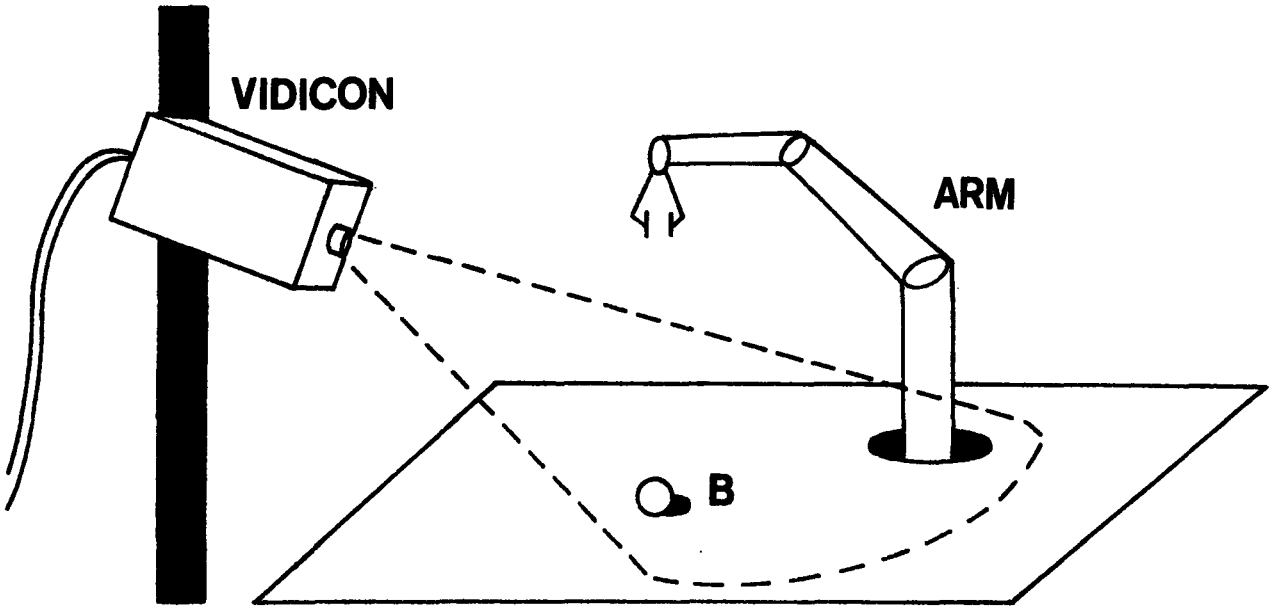

FIGURE 1. Basic set-up. Note arm must be in a non-obscuring angle for the vidicon to view entire field.

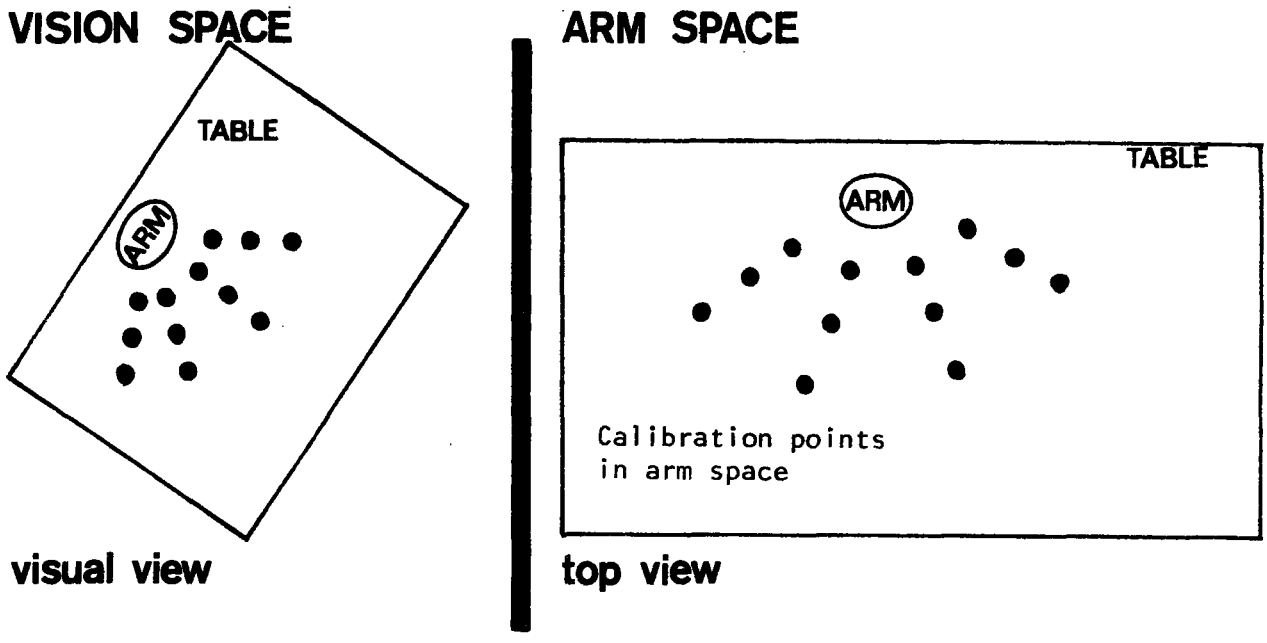

FIGURE 2. Distorted visual view of calibration points.

2 for an example.

This calibration sequence takes from three to five minutes. Once the calibration is done, the vidicon cannot be moved, unless the system is recalibrated. The ball is now placed randomly in the arm's work space (this is done by the arm dropping the ball onto a ramp and letting it roll). The arm again returns to a position out of the way of the visual system, and the ball is located. From the calibration table, the four closest points forming a quadrilateral in vision space about the ball's random location are used to find the balls arm coordinates (see figure 3).

The conversion from vision space to arm space is done as follows:

1) Bisect the sides of the visual quadrilateral.
2) Bisect the sides of the arm space quadrilateral.
3) Decide which quadrant of the visual quadrilateral the point is in.
4) Decide which quadrant in arm space corresponds to that quadrant.
5) Repeat steps 1) through 5) on the new, smaller quadrilateral, UNLESS:
    a) The visual quadrilateral converges to a point, or
        (Return center of remaining arm space quadrilateral)
    b) The arm space quad converges to a point
        (Return this point)

(See figure 4)

It can easily be seen that this method works regardless of how much

the visual system distorts (as long as it always distorts the same), or how inaccurate the arm is in going to a position (again only if it is always inaccurate in the same way), since these errors are incorporated into the calibration table. Note that neither the arm space points nor the visual coordinates for the key points need form a square or rectangle, or any regular figure.

This method is very similar to the bisection method of finding a root of a polynomial, i.e. it will always converge to the point, but slowly. In practice, this system was able to successfully grasp the ping pong ball about 90-95% of the time. The reasons for error were primarily from slight rolling of the ball after being set down during the calibration phase. If the hand detected that its fingers closed too much to have picked up the ball, it backs off, and relocates the ball, and again tries to pick it up. A film of this arm-eye system in operation has been made and can ve viewed at the MIT Artificial Intelligence Labs.

This method generalizes to three dimensions if there are two cameras available. Again, neither the positions, rotations, nor angles of the camera need be known. A three dimensional work area must similarly calibrated by having the arm move to some location, and noting its visual position from each camera. This could be acomplished by attaching some target to the arm and tracking it when it moves, and very accurately finding its position when it is halted. Another method would be one which would be good at recognizing the arm or the target on it, and just move the

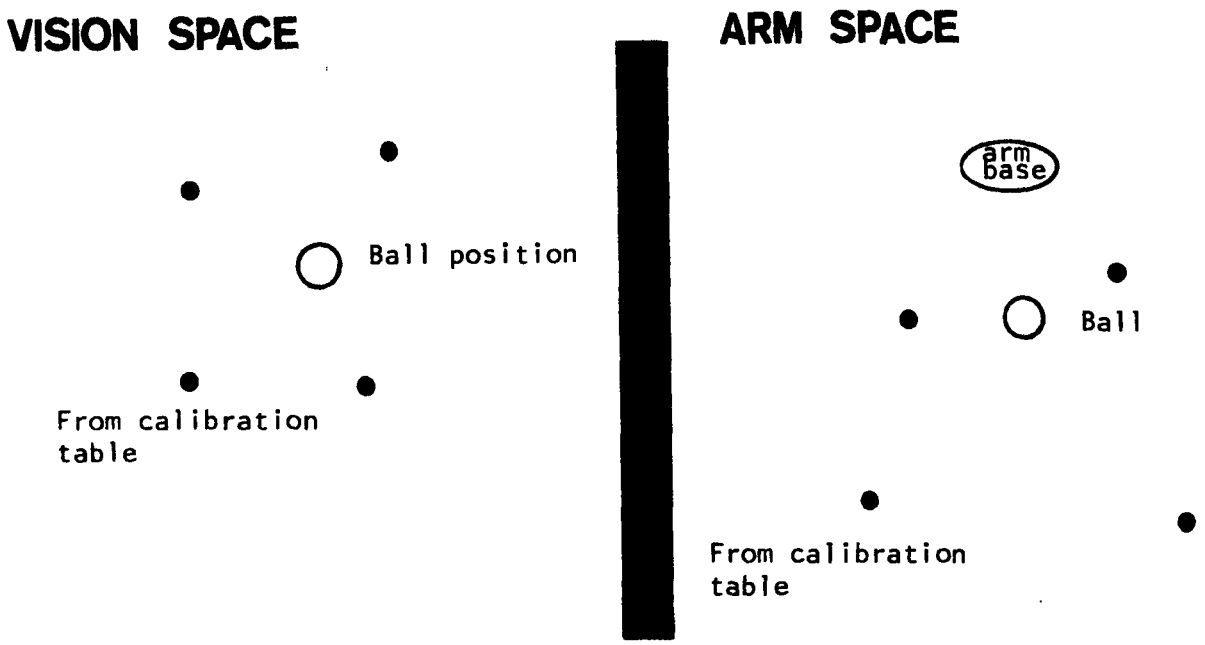

FIGURE 3. Choose four closest calibration points which form a quadralateral about the ball position and their corresponding arm-space coordinates.

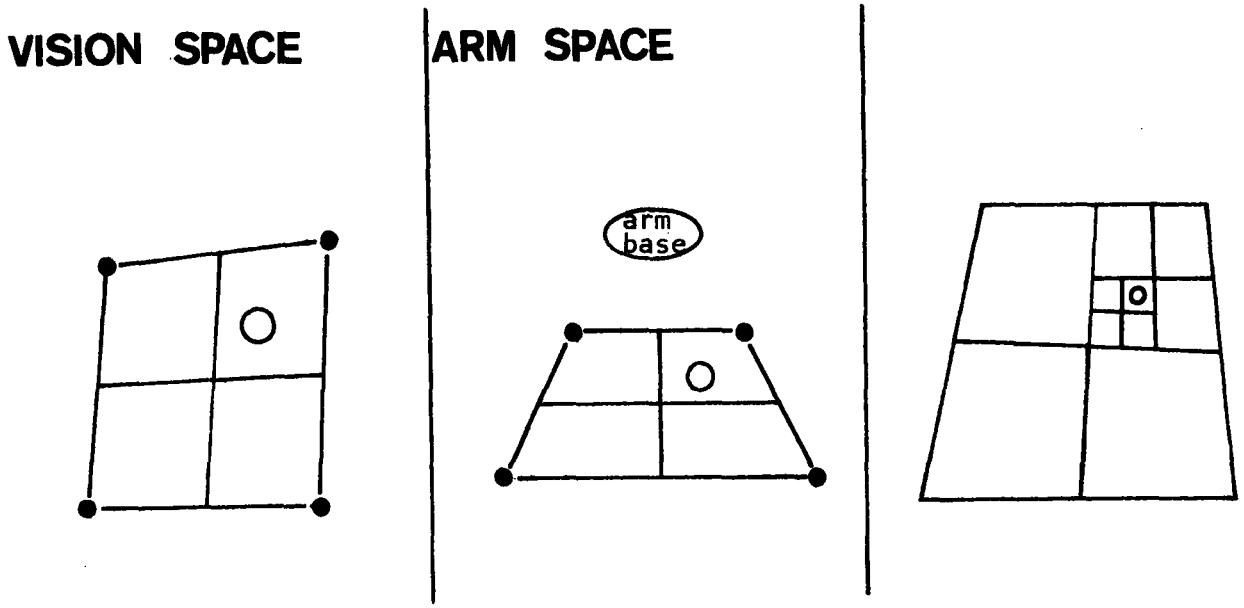

FIGURE 4. Join midpoints of sides of quadralaterals to divide them into four smaller quadralaterals. Find which quadrant ball is in visually and its associated arm-space quadrant. Recurse using this smaller quad.

arm from place to place, calling on the vision to pick the target out of the scene when the arm is halted. Thus this can be treated as a tracking problem or a recognition one.

If the "bottom plane" and the "top plane" (limits of the arm's workspace) are calibrated just as in the one camera method previously described (i.e. four conversion tables, one for each arm and each plane), arbitrary object positions can be found using the following algorithm:

1) Find the ball from eye A, assuming it lies in the top plane.
2) Find the ball from eye A, assuming it lies in the bottom plane.
3) Find the ball from eye B, assuming it lies in the top plane.
4) Find the ball from eye B, assuming it lies in the bottom plane.
5) Calculate the intersection (or point of closest approach) of
   the line through points 1) and 2) and
   the line through points 3) and 4).

While this is a nice generalization, Figure 5 shows that it doesn't cover the entire work area of the arm.

Another generalization to three dimensions is one which geometrically converges on the point without resorting to finding equations of lines. The key to this method is the fact that given the visual and physical coordinates of eight points which form a cube, the visual coordinates of the point to be transformed, and the knowledge that the point lies somewhere in the cube, the cube can be shrunken to one which has

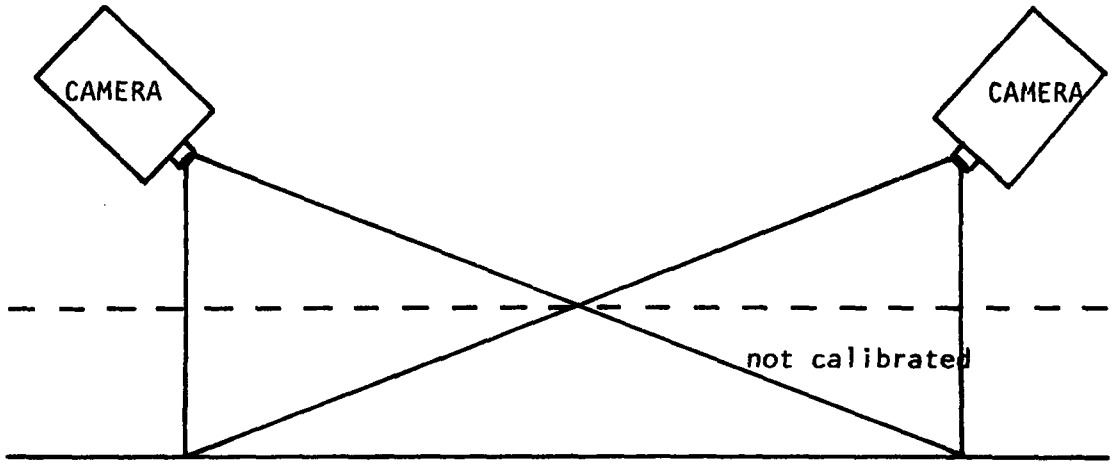

FIGURE 5. Whole workspace not useable.

the point to be transformed in its center visually, i.e. a cube whose diagonally opposite corners lie on the ray from the camera to the point. This shall be called a "corner cube". The idea is to view the point from one camera, eliminate as much of the cube as possible (shrink it to a "corner cube"), then look at the point and shrunken cube from the second camera, and further reduce the cube to one that is a corner cube from that angle. By alternating cameras, this method converges on the point. See figure 6. The way to "shrink a cube" regardless of where the visual image of the point lies is shown in figure 7.

In order to use this method, an initial cube must first be calibrated. Thus some cubical subspace of the arm's working area is the arm-eye working area. If a faster method is desired, the equations of the lines passing through the diagonals of the first two corner cubes can be intersected to

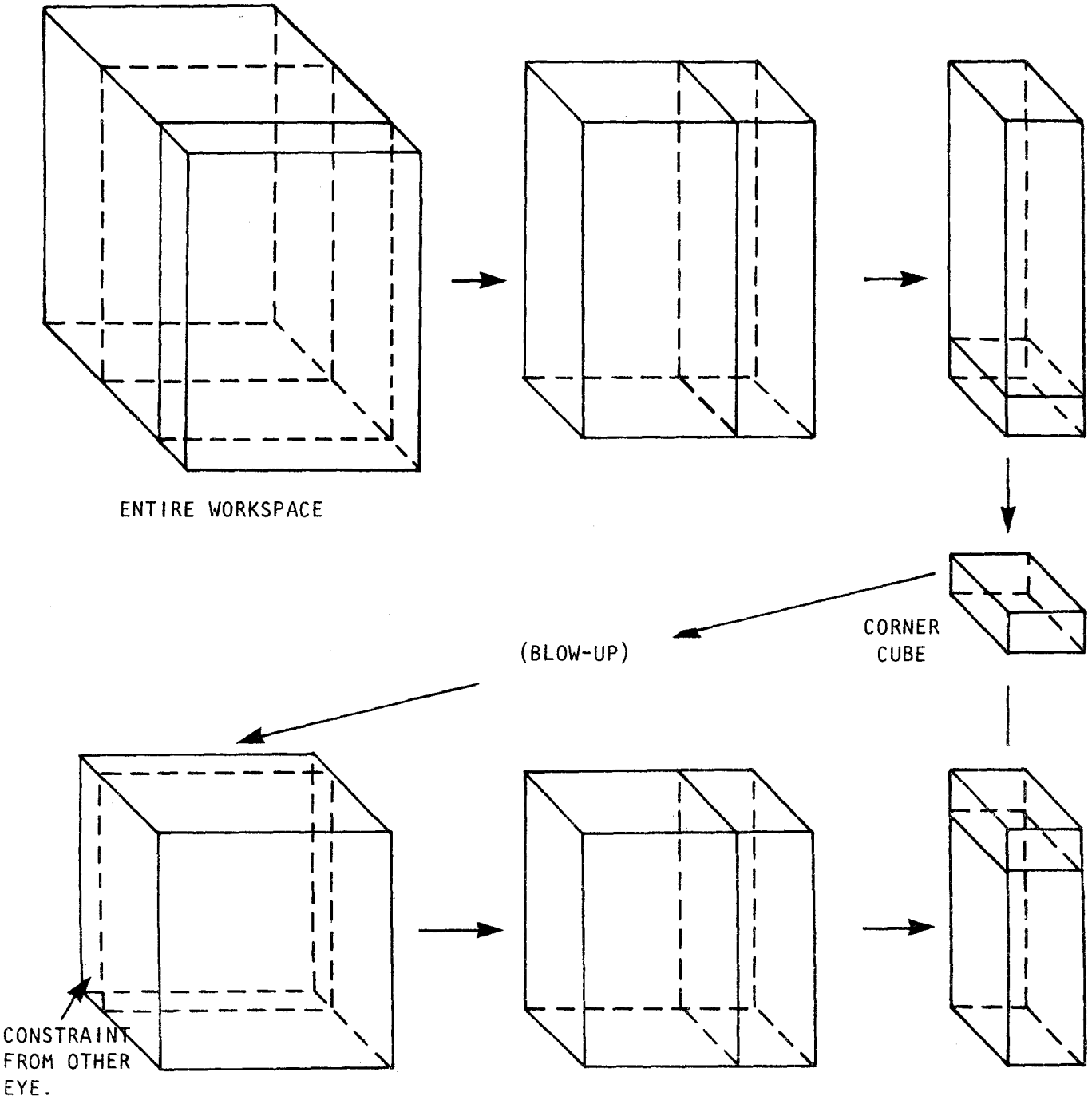

FIGURE 6. Basic geometric method.

find the point. The geometric method has been numerically tested and can be seen to converge quickly at first, but rather slowly near the point, and its only interesting advantage is that is a geometric method of finding the transformation rather than an algebraic one.

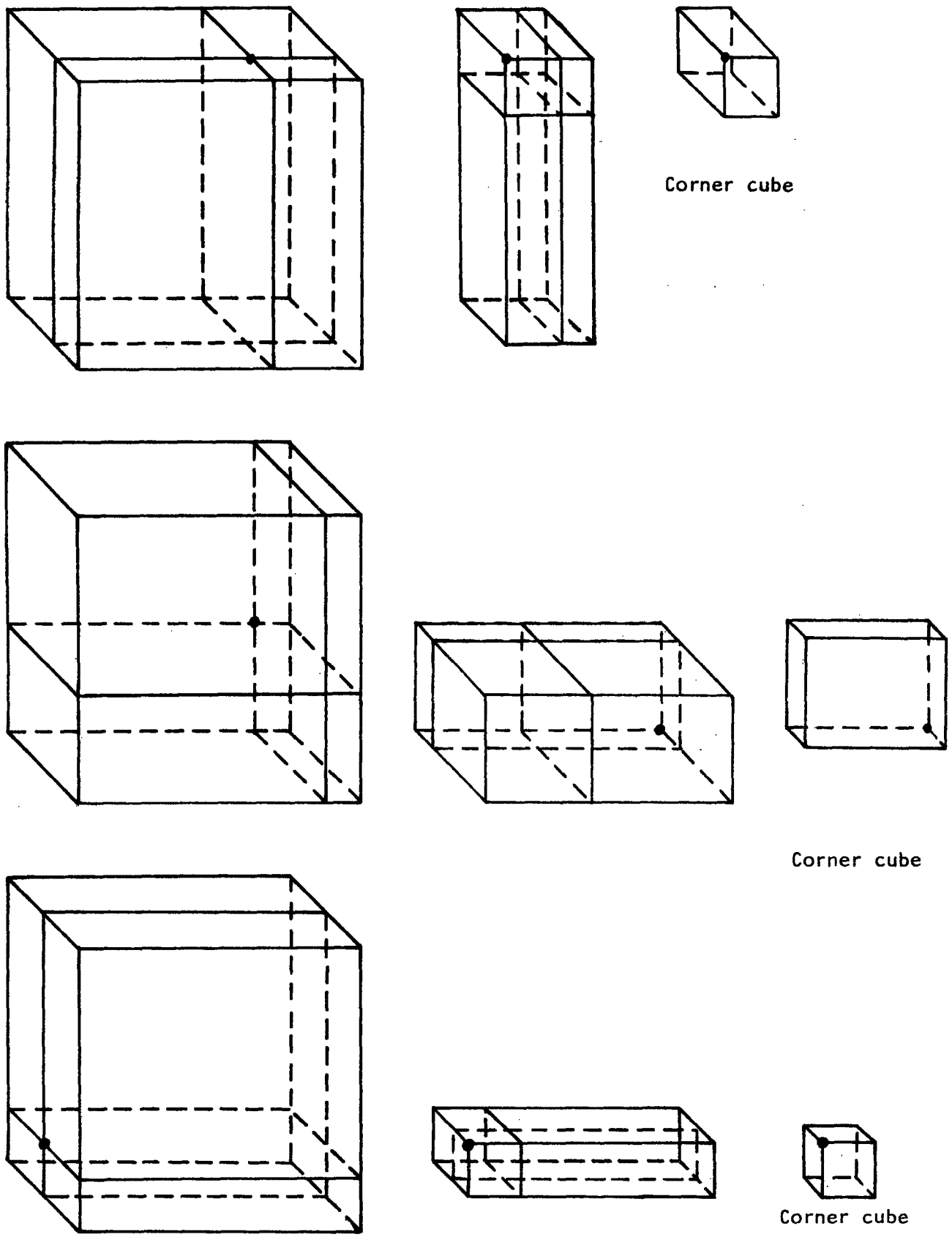

FIGURE 7. All cubes can be reduced to corner cubes independent of position of visual image of point.