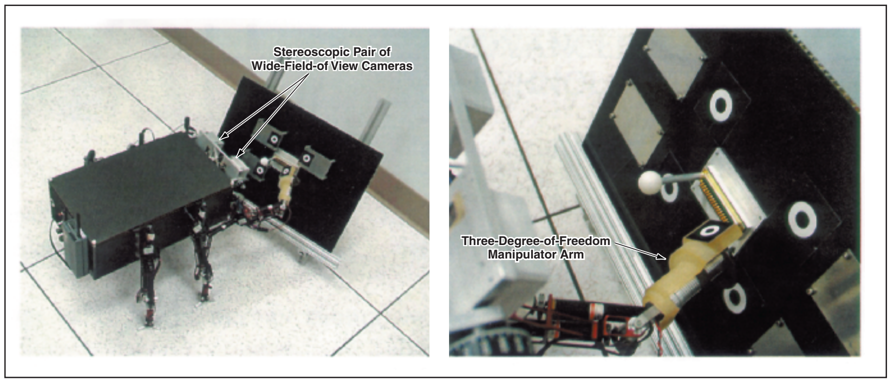
Hybrid Image-Plane/Stereo Manipulation

This method is robust in the face of calibration errors.

NASA's Jet Propulsion Laboratory, Pasadena, California

Hybrid Image-Plane/Stereo (HIPS) manipulation is a method of processing image data, and of controlling a robotic manipulator arm in response to the data, that enables the manipulator arm to place an end-effector (an instrument or tool) precisely with respect to a target (see figure). Unlike other stereoscopic machine-vision-based methods of controlling robots, this method is robust in the face of calibration errors and changes in calibration during operation.

In this method, a stereoscopic pair of cameras on the robot first acquires images of the manipulator at a set of predefined poses. The image data are processed to obtain image-plane coordinates of known visible features of the end-effector. Next, there is computed an initial calibration in the form of a mapping between (1) the image-plane coordinates and (2) the nominal three-dimensional coordinates of the noted end-effector features in a reference frame fixed to the main robot body at the base of the manipulator. The nominal three-dimensional coordinates are obtained by use of the nominal forward kinematics of the manipulator arm — that is, calculated by use of the currently measured manipulator joint angles and previously measured lengths of manipulator arm segments under the assumption that the arm segments are rigid, that the arm lengths are constant, and that there is no backlash. It is understood from the outset that these nominal three-dimensional coordinates are likely to contain possibly significant calibration errors, but the effects of the errors are progressively reduced, as described next.


Stereoscopic Pair of Wide-Field-of View Cameras
Three-Degree-of-Freedom Manipulator Arm

A **Stereoscopic Pair of Electronic Cameras** on a legged mobile exploratory robot acquires images of a scene that includes a manipulator arm and a target. The image data are used to guide the end effector of the manipulator toward the target.

As the end-effector is moved toward the target, the calibration is updated repeatedly by use of data from newly acquired images of the end-effector and of the corresponding nominal coordinates in the manipulator reference frame. By use of the updated calibration, the coordinates of the target are computed in manipulator-reference-frame coordinates and then used to the necessary manipulator joint angles to position and orient the end-effector at the target with respect to the same kinematic model from the calibration step. As the end-effector/target distance decreases, the computed coordinates of the end-effector and target become more nearly affected by the same errors, so that the differences between their coordinates become increasingly precise. When the end-effector reaches the target, the remaining effective position error is the distance that corresponds to more than about one pixel in the stereoscopic images of the target.

This work was done by Eric Baumgartner and Matthew Robinson of Caltech for **NASA's Jet Propulsion Laboratory**. *Further information is contained in a TSP (see page 1).*
NPO-30492

Partitioning a Gridded Rectangle Into Smaller Rectangles

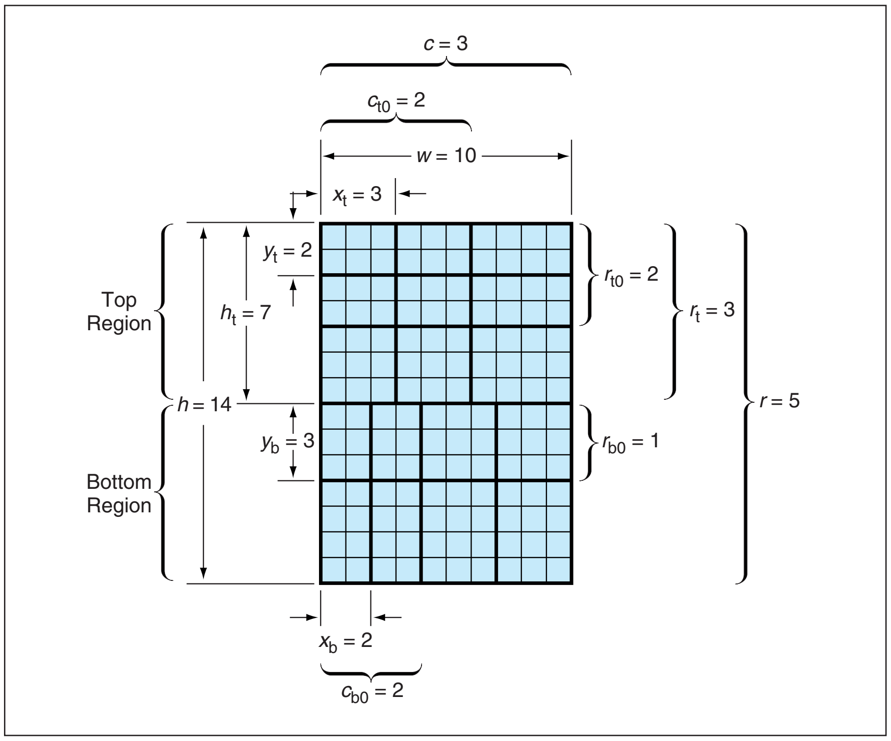
A relatively simple algorithm yields nearly square, nearly equally sized segments.

NASA's Jet Propulsion Laboratory, Pasadena, California

A relatively simple algorithm, devised for use in an image-data-compression application, partitions a rectangular pixelated image (or any other rectangle on which a regular rectangular grid has already been superimposed) into a specified number of smaller rectangles, hereafter denoted segments. The algorithm has the following properties:

- No floating-point operations are needed.
- The segments tend to be nearly square (in the sense that their widths and heights in pixel units tend to be nearly equal).
- The segments tend to have nearly equal areas.
- The algorithm yields valid results (no zero-width or zero-height segments) as long as the specified number of segments, *s*, does not exceed the number of pixels (equivalently, the number of grid cells).

The inputs to the algorithm are the positive integer *s* plus the positive integers *h* and *w*, denoting the height and width, respectively, of the rectangle in



A **Rectangular Image** of 10 by 14 pixels is partitioned into 17 segments by the algorithm described in the text.

pixel units. The limit on s for a valid result is given by $s \le wh$.

The output of the algorithm is characterized by, and can be described completely in terms of, several parameters that are illustrated by the example shown in the figure. The segments are arranged in r rows. A top region of the rectangle contains segments arranged in c columns. A possible bottom region contains segments arranged in $c + 1$ columns. The top region has height h_t and contains r_t rows of segments. The first r_{t0} rows of segments in the top region have height y_t; the remaining rows (if any) in the top region have height $y_t + 1$. The first c_{t0} columns in the top region have width x_t; any remaining columns in the top region have width $x_t + 1$. Similarly, the first r_{b0} rows in the bottom region have height y_b, and the remaining rows in the bottom region have height $y_b + 1$, while the first c_{b0} columns in the bottom region have width x_b and the remaining columns in the bottom region have width $x_b + 1$.

The steps of the algorithm are the following: First, r is computed. If $h > (s-1)w$ then $r = s$. Otherwise, r is the unique positive integer that satisfies

$$(r-1)rw < hs \le (r+1)rw.$$

Some of the other parameters are computed as follows:

$$c = \left\lfloor \frac{s}{r} \right\rfloor$$
$$r_t = (c+1)r - s$$
$$h_t = \max\left(r_t, \left\lfloor \frac{hcr_t}{s} + \frac{1}{2} \right\rfloor\right)$$
$$x_t = \left\lfloor \frac{w}{c} \right\rfloor$$
$$c_{t0} = (x_t + 1)c - w$$
$$y_t = \left\lfloor \frac{h_t}{r_t} \right\rfloor$$
$$r_{t0} = (y_t + 1)r_t - h_t.$$

If $r_t < r$, so that there is a bottom region, then the remaining parameters are computed as follows:

$$x_b = \left\lfloor \frac{w}{c+1} \right\rfloor$$
$$c_{b0} = (x_b + 1)(c+1) - w$$
$$y_b = \left\lfloor \frac{h - h_t}{r - r_t} \right\rfloor$$
$$r_{b0} = (y_b + 1)(r - r_t + 1) - (h - h_t).$$

It has been verified by straightforward algebraic analysis of these equations that the algorithm has the properties mentioned in the first paragraph.

This work was done by Matthew Klimesh and Aaron Kiely of Caltech for **NASA's Jet Propulsion Laboratory**. *Further information is contained in a TSP (see page 1).*

This software is available for commercial licensing. Please contact Don Hart of the California Institute of Technology at (818) 393-3425. Refer to NPO-30479.

Digital Radar-Signal Processors Implemented in FPGAs

Processing can be performed onboard at relatively low power.

NASA's Jet Propulsion Laboratory, Pasadena, California

High-performance digital electronic circuits for onboard processing of return signals in an airborne precipitation-measuring radar system have been implemented in commercially available field-programmable gate arrays (FPGAs). Previously, it was standard practice to downlink the radar-return data to a ground station for postprocessing — a costly practice that prevents the nearly-real-time use of the data for automated targeting. In principle, the onboard processing could be performed by a system of about 20 personal-computer-type microprocessors; relative to such a system, the present FPGA-based processor is much smaller and consumes much less power. Alternatively, the onboard processing could be performed by an application-specific integrated circuit (ASIC), but in comparison with an ASIC implementation, the present FPGA implementation offers the advantages of (1) greater flexibility for research applications like the present one and (2) lower cost in the small production volumes typical of research applications.

The generation and processing of signals in the airborne precipitation-