Publications of the Astronomical Society of the Pacific
107: 279–288, 1995 March

Decision Trees for Automated Identification of Cosmic-Ray Hits in *Hubble Space Telescope* Images

STEVEN SALZBERG[1]

Department of Computer Science, Johns Hopkins University, Baltimore, Maryland 21218
Electronic mail: salzberg@cs.jhu.edu

RUPALI CHANDAR

Department of Physics and Astronomy, Johns Hopkins University, Baltimore, Maryland 21218
Electronic mail: rupali@jhufos.pha.jhu.edu

HOLLAND FORD

Department of Physics and Astronomy, Johns Hopkins University, Baltimore, Maryland 21218
Electronic mail: ford@stsci.edu

SREERAMA K. MURTHY

Department of Computer Science, Johns Hopkins University, Baltimore, Maryland 21218
Electronic mail: murthy@cs.jhu.edu

RICHARD WHITE

Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, Maryland 21218
Electronic mail: rlw@stsci.edu

Received 1994 April 18; accepted 1994 November 30

ABSTRACT. We have developed several algorithms for classifying objects in astronomical images. These algorithms have been used to label stars, galaxies, cosmic rays, plate defects, and other types of objects in sky surveys and other image databases. Our primary goal has been to develop techniques that classify with high accuracy, in order to ensure that celestial objects are not stored in the wrong catalogs. In addition, classification time must be fast due to the large number of classifications and to future needs for on-line classification systems. This paper reports on our results from using decision-tree classifiers to identify cosmic-ray hits in *Hubble Space Telescope* images. This method produces classifiers with over 95% accuracy using data from a single, unpaired image. Our experiments indicate that this accuracy will get even higher if methods for eliminating background noise improve.

1. INTRODUCTION

We have been developing new methods to address the goal of providing the astrophysical research community with tools for analyzing very large, complex, multiparameter data sets. This paper describes a joint effort by researchers in artificial intelligence (AI) and astrophysics to develop and apply computational techniques to significant problems in astronomy. Specifically, we are using several techniques from the machine learning subfield of AI in order to identify and classify objects in astronomical images. This paper describes the results of an ongoing effort to identify accurately the types of noise commonly present in *Hubble Space Telescope* (*HST*) data, especially cosmic-ray hits, which are very common in Wide Field Camera images. The main techniques to be described below are algorithms for building decision trees. These trees have proven to be fast, accurate classifiers for numerous data sets, and our hope was that they would work well for the *HST*/WFC cosmic-ray problem. Our results and comparisons with other machine learning techniques show that decision trees are excellent classifiers for many types of problems. Their accuracy on the cosmic-ray problem is currently about 95%. In order to improve this figure, astronomers may have to develop better methods for identifying and removing spurious sources such as hot pixels from *HST* images.

Below we describe, first, the star/cosmic-ray classification problem for *HST* images. We then describe our decision-tree algorithm, as well as some related techniques, and present the results of our experiments with these methods.

2. CLASSIFICATION OF STARS AND COSMIC-RAY HITS IN IMAGES OBTAINED WITH THE WIDE FIELD CAMERA IN THE *HUBBLE SPACE TELESCOPE*

Cosmic-ray (CR) primaries and secondaries striking the CCD detectors in the *HST*'s first Wide Field and Planetary Camera (WF/PC-1) created electron-hole pairs in the silicon that were detected along with the electron-hole pairs created by the photons striking the CCD. During a typical 20-min exposure, each of the four CCDs detected approximately 2000 CR events. Many of the low-amplitude CR events look like faint stars. We have been using our classification techniques to learn to identify CRs and stars in *HST* data; our first goal was to find classifiers that separated CRs and stars

[1]Direct correspondence to Salzberg.

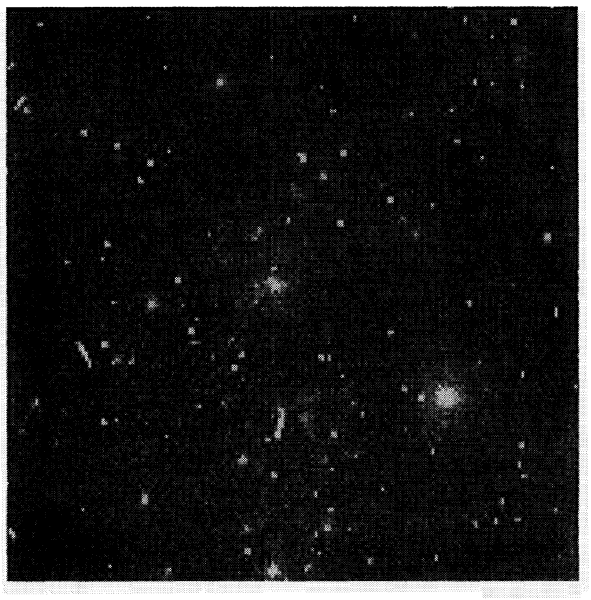

FIG. 1—A portion of a 900 s CCD WF1 image of M81 taken through a yellow filter (F555W), containing stars, cosmic rays, and other sources of CCD and sky noise.

with high reliability in *individual*, aberrated images obtained with the WF/PC. Progress toward that goal is described below. Our long-range goal, discussed in Sec. 5, is to extend the present successful classification method to the aberration-corrected images, which are being obtained with the new Wide Field Planetary Camera (WFPC2) that was installed in the *HST* by the crew of the Space Shuttle Endeavor in 1993 December.

We began our study by selecting two images from an *HST* "Key Project" aimed at measuring the Hubble Constant, i.e., the rate at which the universe is expanding. The images are two 900 s WFC exposures through a yellow filter ($F555W$) of a field in the nearby galaxy M81 taken at two different times. Figure 1 shows part of an image from one of the four CCDs (CCD WF1) that were exposed simultaneously when a WF/PC picture was taken. The entire field, which was centered on a portion of a spiral arm in M81, contains thousands of faint stars. Because the stars are faint and close to one another, the image is a difficult test case.

Our first objective was to create a highly reliable catalog of the positions and identities of stars and cosmic rays. We used a standard program (the task "Combine" in the Space Telescope Institute's software package STSDAS) with one-sided, three sigma cutoff to combine the two images of the M81 field. The program retains objects (stars in this case) that are statistically above the noise in each image, and rejects objects (CRs) that appear in only one image. Note that our goal is to eliminate the need for split exposures, and thereby save the significant amount of spacecraft time that is required to prepare the WF/PC for the second exposure. We also wish to eliminate the noise penalty that is incurred by taking two exposures. The quadratic sum of the root mean square (rms) read noise and preflash noise in CCD WF1 was ~20 electrons per pixel. The rms read noise in WFPC2 is ~5.2 electrons per pixel. Whenever the exposure is split and then subsequently summed after CR removal, the read noise is increased by $\sqrt{2}$. This noise competes with the noise in the sky-background in short exposures or in exposures made through narrow-band filters. We next used the program DAOFIND to catalog the (x,y) positions of the stars in the image. The mean background of the combined image varied from 27 to 33 counts in WF1, with an average of 31 counts. The standard deviation varied from 2.1 to 2.8, with an average of 2.55. Similar means and standard deviations were found for the other three CCDs. The threshold used for object detection by DAOFIND was 9 counts above the background, which along with the 2.55 standard deviation of the background gives a 3.5σ detection limit. By subtracting the combined image from each of the individual images, we obtained two images which contain only CRs. We then used DAOFIND on each of the CR images to catalog the positions of the CRs.

As an external check on the completeness of this CR catalogue, we found the surface density of the CRs and compared it to the values recently determined in (Windhorst et al. 1994). The Final Orbital/Science Verification Report by the WF/PC-1 Investigation Definition Team lists zeropoint offsets for the M81 field for WF2 (Hunter et al. 1991). Consequently, we used the same CCD for our CR surface density determination. We defined a cosmic ray to be a single pixel in the CR image with a threshold of $4.0\times\sigma_{\rm local}$ in ADU flux. A fairly high threshold was used because the noise in the two input images was heavily correlated, leading to an artificially low rms noise in the difference (CR) image. The (x,y) catalog of these CR positions on WF2 was used as input to the IRAF task DAOPHOT.PHOT, which performed 0.55 pixel aperture photometry. This essentially gives the counts in a single pixel, which is appropriate for cosmic rays. The V magnitude given by PHOT treats CRs as if they were real objects on the sky, and is defined as $C_v-2.5\log$ ADU, where ADU is the ADU flux a CR would have generated in the image during the total exposure time. We histogramed the V magnitudes from both images.

Before plotting the histogram, we needed to find the correct zero-point magnitude for our data, in order to use it as an offset. We followed a procedure similar to that outlined in

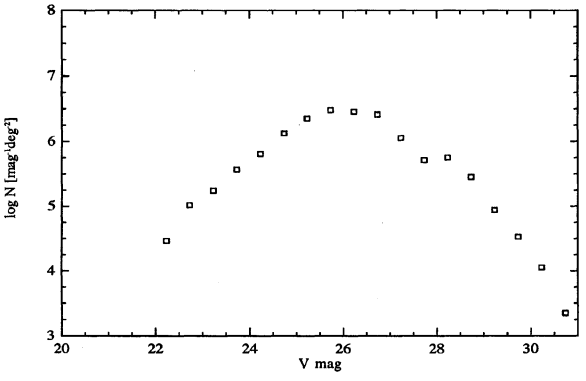

FIG. 2—*HST*/WFC cosmic-ray "counts" ($F555W$).

Freedman et al. (1994). We located the five Cepheids listed for CCD WF2 and did core fitting with an aperture of 2.5 pixels. In order to perform the aperture correction, we used a theoretical PSF generated by TinyTim for WF2. From this we found that ~14.3% of the light falls within a 2.5 pixel radius. We ignored color corrections as was done in (Hunter et al. 1991), but we did perform aperture corrections. Comparing our F555W V magnitudes to the V magnitudes listed by the IDT, we confirmed the zeropoint magnitude of 23.0 to within 0.1 mag (or 10%), and used this number as our offset. Figure 2 plots the (log of the) number of pixels affected by cosmic rays as a function of V magnitude. The peak of our CR surface density occurs around magnitude 26, whereas it is around magnitude 27 in Windhorst et al.'s paper. This difference is most likely due to the much longer exposure times used in Windhorst et al.'s paper. We find that the differential CR "magnitude counts" closely follow an apparent power law, down to $V \simeq 27$ magnitude, with a slope $\gamma = 0.57$ [when counted as $N(m) \propto m^{\gamma}$], compared to $\gamma = 0.6$ in Windhorst et al.'s paper. Our observed surface density of CR hits is 3.96 pix s^{-1} CCD. This corresponds to ~3 events $cm^{-2}\ s^{-1}$. (Note that we have defined a CR event to consist of one pixel which has signal above a certain threshold in the image containing only CRs.) In general, however, our results are consistent with theirs.

Any classification method uses a set of *features* to characterize each object; obviously, the features should be tailored to the task at hand. We took two approaches to extracting features to be used for classification. The first approach has the appeal of maximum simplicity: simply give the classifier the raw data. A 3×3 array of pixels centered on a faint star contains most of the information about the star. Consequently, we extracted a 3×3 "postage stamp" centered on each star and each CR. Using the nine intensity values from this array as our features, each star and CR becomes a point in a nine-dimensional space. If the stars and CRs form separate clusters in this space, and if the classifier can separate out those clusters, we will be able to classify the objects accurately. Our second approach is in principle more powerful. We know in great detail how aberration and diffraction determined the *HST* intensity distribution from point sources (i.e., stars), the so-called point-spread function or PSF, in WF/PC-1 images. If we fit a PSF to the stars and CRs, the parameters of the fit should be very different for the two classes. Details of these parameters are given in Appendix A.

3. CONSTRUCTING A DECISION-TREE CLASSIFIER

Using these different parameters, we constructed a decision tree using our new OC1 system and several other well-known classification methods. OC1 (**O**blique **C**lassifier 1) builds decision trees that contain an oblique (not axis-parallel) hyperplane at each node. We briefly describe the OC1 algorithm below; for more details see Murthy et al. (1994).

A decision tree is a data structure that contains tests at its internal nodes and class labels at the leaf nodes. A simple 2-D example is illustrated in Fig. 3. A decision tree is used as

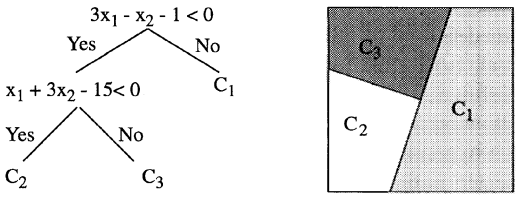

FIG. 3—The left side shows a small "oblique" decision tree that uses only two attributes. There are two internal nodes and three leaf nodes in this tree. The right side shows how this tree partitions the 2D attribute space.

a classifier by passing an example down through the tree beginning at the root (topmost) node. Each internal node contains a test of the form

$$\sum_{i=1}^{d} (a_i X_i) + a_{d+1} > 0, \qquad (1)$$

where each example X has d attributes or features. Clearly, this test just measures whether each example is above or below a d-dimensional hyperplane defined by the a_i's. A node may also contain a simpler test of the form $a_i X_i > k$; i.e., an axis-parallel hyperplane. If the test is true, the example is sent down the "yes" branch of the tree, and otherwise it goes down the "no" branch. This process continues until the example reaches a leaf node. The leaf nodes of the tree contain category labels, in this case either "star" or "cosmic ray."

Many decision-tree methods have been used for classification (Moret 1982), most notably C4.5 (Quinlan 1993) and CART (Breiman et al. 1984). These classifiers were included in our experiments as well, since our goal was to find the best overall method for the star/cosmic-ray problem. As an additional baseline for our comparison, we included tests using the nearest-neighbor algorithm, which has a long history of use for pattern recognition and classification problems (Dasarathy 1991). The principle difference between OC1 and other decision-tree algorithms is the form of the test at each node: these other methods use only axis-parallel hyperplanes, while OC1 considers both axis-parallel and oblique hyperplanes. The simpler tests sometimes perform very well, but their performance depends on the concept or function that the system is attempting to learn, as well as the set of features used to represent the examples. One advantage of axis-parallel tests is that *all* tests of this form can be considered as the tree is being built. For a data set with n examples and d features, there are only $d \cdot (n-1)$ distinct axis-parallel tests (i.e., hyperplanes that divide the examples into distinct subsets). The number of distinct oblique hyperplanes, however, is $2^d \cdot \binom{n}{d}$; and in fact the problem of finding the optimal oblique split is NP-Complete.[2] OC1 therefore takes a heuristic approach and uses *local search* augmented with

[2]An NP-Completeness result implies that it is extremely unlikely that any method could solve the problem efficiently; i.e., using time that is a polynomial function of n and d. Informally stated, no algorithm is guaranteed to find an optimal hyperplane unless it uses an exponential amount of time. See Garey and Johnson (1979) for a detailed introduction to NP-Completeness. The NP-Completeness proof for an oblique hyperplane is due to Heath (1992).

randomization to find good tests at each node, as will be explained below. In other extensive empirical studies (Murthy et al. 1993,1994), we found that this approach was very successful at finding small, accurate decision-tree classifiers.

Following is an outline of the algorithm that OC1 uses to select a hyperplane at each node of the tree.[3] This procedure is repeated until the tree correctly classifies all the training examples, after which the tree is pruned back to avoid overfitting (i.e., fitting the noise in the data).

OC1 and CART prune their trees using Breiman et al.'s Cost Complexity pruning, which reserves part of the training set as a separate pruning set. Once a complete tree has been induced, both systems execute the following procedure until only the root of the tree is left: determine the "weakest" subtree and remove it. The general idea is that after removing any part of the tree, one can then recompute how well the tree that remains classifies the training set. The weakest subtree is the one whose removal has the smallest effect on this measure [for details, including a formal definition of "weakest" and of how accuracy is estimated for the training set, see Breiman et al. (1984)]. This process results in a set of successively smaller trees, and OC1 chooses the tree from this set that gives maximum accuracy on the separate pruning set. The user can adjust the size of this pruning set, or even turn off pruning entirely.

3.1 Picking a Good Split

In order to choose a good hyperplane as a test at a decision-tree node, we need to have a measure of goodness. Many such measures have been defined in the literature, and one that has worked well for both OC1 and CART is the "twoing" criterion (Breiman et al. 1984). This criterion assigns higher goodness values to hyperplanes that come close to splitting the data in half, and to those that do not split up examples from the same class. In other words, the ideal split will have all examples from class c_j on the same side of the hyperplane, and will put exactly half the total number of examples on each side. This measure works well for two-class and also multi-class problems. If we denote the proportion of examples on the left side of a split as p_L and the proportion on the right as p_R, then the twoing criterion is defined as

$$(p_L * p_R)\left(\sum_j \left|[p(j|L)-p(j|R)]\right|\right)^2, \qquad (2)$$

where $p(j|L)$ and $p(j|R)$ are the proportions of class j on the left and right sides of the split.

The CART algorithm picks the axis-parallel hyperplane that maximizes this criterion as the test at a decision-tree node. OC1 considers both axis-parallel and oblique splits, and picks an oblique hyperplane when it finds one that has a higher twoing criterion value than the best axis-parallel split.[4]

3.2 Searching for a Good Oblique Hyperplane

To find a good oblique hyperplane, OC1 first selects a *random* initial location at each node of the tree. Even if such a randomly placed hyperplane has a very poor location, subsequent perturbations should quickly improve it.

The strategy of searching through the space of possible hyperplanes is defined by the procedure that perturbs the current hyperplane into a new location. First let P be the set of n examples at a given node of the tree. The equation of the current hyperplane H can be written as

$$\sum_{i=1}^{d} (a_i X_i) + a_{d+1} = 0. \qquad (3)$$

We can plug each example into this equation to determine quickly whether it is above or below the hyperplane. Let $P_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$ be the jth example from the training set. If we substitute P_j into the Eq. 3, we get: $\sum_{i=1}^{d}(a_i x_{ji}) + a_{d+1} = V_j$, where $V_j > 0$ only if point P_j is above the hyperplane H. If H splits the training set perfectly, then all points belonging to the same category will have the same sign; i.e., $sign(V_j) = sign(V_k)$ if and only if $category(P_i) = category(P_j)$.

OC1 then considers adjusting the coefficients of H one at a time. By treating a single coefficient a_m as a variable, and all other coefficients as constants, we can compute the optimal value for a_m, using an idea from Breiman et al. (1984). OC1 cycles through the d coefficients until it can no longer improve any of them; i.e., it has reached a locally optimal hyperplane.

It is natural to ask why we should only consider linear tests—why not have much more general tests, such as kth degree polynomials at each node of a decision tree? The principal barrier to considering such tests is the enormous increase in the computational cost of searching for such polynomials. While the number of distinct hyperplanes is already very large—$2^d \cdot \binom{n}{d}$—the number of other types of surfaces is much greater.

3.3 Randomization to Escape Local Maxima

A significant problem in searching for the best hyperplane (and in many other optimization problems) is that of local maxima. The search process is said to have reached a local

[3]The complete OC1 package is available over the Internet via ftp. This contains documentation describing the pruning procedures, goodness measures, and other parameters that the user can adjust. To obtain instructions on how to retrieve the package, send a request to salzberg@cs.jhu.edu or murthy@cs.jhu.edu.

[4]The best oblique split of a set of points is always at least as good as the best axis-parallel split. But as mentioned earlier, finding the best oblique split is computationally infeasible. OC1's heuristics allow it to find good splits, but in some cases it is possible that these will be worse than the best axis-parallel split. Even when an oblique split is slightly better than the best axis-parallel split, the user may prefer to use the axis-parallel split because it has fewer free parameters. This preference can be supplied as an input parameter to OC1.

maximum if no adjustment of the current hyperplane, as suggested by the perturbation algorithm, improves the goodness measure.

We have implemented two ways of escaping from local maxima. The first one is easy: simply restart the search from a new random location. Because the algorithm is fast, it can easily consider 20–100 random starting points at each node.

The second randomization step involves perturbing the hyperplane in a random direction after it reaches a local maximum. We do this by choosing a random direction represented by a vector $R=(r_1,r_2,\ldots,r_d)$. We then add αR to the coefficient vector A, after first computing the optimal value for α. [For details see Murthy et al. (1994).] OC1 will consider a small constant number (determined by the user) of these "random jumps" each time it gets stuck. Note that although moving the hyperplane in a random direction modifies all the coefficients at once, it only increases the time complexity of the algorithm by a constant factor.

We should emphasize that randomization as used here is only a heuristic that works well when OC1 is trying to find the best hyperplane. Neither this heuristic nor any other is guaranteed to find the global maximum; the NP-Completeness of the problem implies that no such heuristic exists.

3.4 Estimating Accuracy of a Tree

One measure of the accuracy of a decision tree is how well it is able to classify the data provided to it, called the *training set*. For the purposes of this study and most others, we assume that the class labels of the examples in the training set are correct. Of course some of these labels may be wrong, in which case the decision tree will not be as accurate as our estimates. This highlights the importance of generating good training sets.

Naturally, one can always build a tree that achieves 100% accuracy on the training set. Usually, however, we would like to know how accurate a tree will be at classifying other, *unseen* objects drawn from the same distribution as the training set. To estimate this accuracy, a standard practice is to reserve a portion of the training data as a separate *test* set, which is not used in building the tree. The accuracy of the tree on this test set is then used as an estimate of the accuracy for unseen examples.

In some of our experiments, we used a well-known statistical technique called cross-validation to estimate accuracy. This is described below in Sec. 4.1. As a further check on these accuracy figures, we collected additional data from a different CCD and used a tree built on the first CCD to classify this data. This provided us another estimate of accuracy.

In all experiments, we report three accuracy figures: overall, stars, and cosmic rays. Overall accuracy is just the percentage of correct classifications over the whole test set, where "correct" means the decision tree agreed with the class label provided with the input data. Accuracy on stars (and respectively cosmic rays) is the percentage of the time that a prediction of "star" was correct. In other words, if the system predicted "star" 100 times, and the class label in the data was "star" for 95 of those predictions, then accuracy would be 95% for stars.

4. EXPERIMENTAL DESIGN AND RESULTS

Each object was identified by a set of predefined features. For our preliminary experiments, we used the simplest possible object definition: simply give the classifier the raw (calibrated) data. As mentioned above, this consisted of the nine intensity values from a 3×3 array of pixels centered on each star and each CR. After some preliminary trials, we determined that we could improve our results substantially by using additional features relevant to the problem. We know in great detail how diffraction and aberration in (pre-servicing mission) *HST* images determine the intensity distribution from point sources (i.e., stars). We fit a PSF to the star and CR images, and used parameters from that plus other knowledge about the differences between stars and CRs to define 11 additional features. This gave us a total of 20 parameters, described in Appendix A. All the algorithms tested were given the full set of 20 parameters.

A relatively small number of cosmic rays (287 out of 4689 objects) were superposed on the images of stars. These *blends* (stars plus CRs) are a third class of objects that we wish to classify at a later date when we have a sufficient number of examples. We removed these blends from our database for the experiments described below. The database consisted of two sets of objects, taken from two adjacent CCDs, WF1 and WF2, from a 4-CCD array. WF1 was used to develop the decision-tree classifiers, and give initial estimates of accuracy. We reserved WF2 to provide an independent test of the accuracy of each classifier; the idea was to use a decision tree induced from WF1 data to classify WF2 data. We chose adjacent CCDs in order to equalize global conditions such as background sky brightness as much as possible. Including blends, WF1 contained 2430 objects and WF2 contained 2484 objects. After removing the blends, WF1 contained 2259 objects and WF2 contained 2368 objects. In both cases about 60% of the objects were cosmic rays.

4.1 Experiments on WF1 Data

To estimate the accuracy of a classifier on the WF1 data, we ran a five-fold *cross-validation* experiment as follows. We divided the data into five equal-sized sets. Then for each of the five subsets, we used the remaining $\frac{4}{5}$ of the data as a training set and reserved the fifth set as a test set. Thus each object belonged to exactly one test set, and the accuracy measure is the percentage of correct classifications over the entire data set. (Reminder: "correct" means the decision tree agreed with the class label provided with the data. If that label is incorrect, the algorithm has no way of knowing.) Cross-validation studies are well known to provide good estimates of accuracy, with relatively little "optimistic" bias.

We ran four different decision-tree algorithms on this data: CART (Breiman et al. 1984), C4.5 (Quinlan 1993), and two versions of OC1. In addition to the version of OC1 described above, we also used a version that only considers axis-parallel (univariate) tests at each node of the tree. We

TABLE 1
Comparison of Decision-Tree Accuracies for WF1 Data

	Accuracy (%)			Tree
Algorithm	Overall	Stars	Cosmic Rays	Size
CART	93.7±1.4	90.2	95.7	6.6±2.3
C4.5	92.7±1.3	89.1	94.8	36.6±3.1
OC1	91.8±1.2	87.9	94.1	7.5±1.3
OC1-AP	92.9±0.5	88.6	95.4	7.3±1.4
5-NN	89.4	87.3	90.6	

call this version OC1-AP.[5] In addition, we used the five-nearest-neighbor (5-NN) classification method. 5-NN simply stores all examples in the training set, and classifies test examples by finding the five nearest (using Euclidean distance in the feature space) examples from the stored set. The majority vote among the nearest neighbors determines the class. Because this method is known to perform well on a very wide class of problems, the 5-NN results serve as a kind of baseline.

Each decision-tree system was run using the default settings supplied with those systems. Consequently, the goodness measures and pruning techniques used are different for each of the methods. Detailed descriptions of the default settings and what they mean for each system can be found in the following references: C4.5 (Quinlan 1993), CART (Breiman et al. 1984), and OC1 (Murthy et al. 1994).

The accuracies on WF1 are given in Table 1. Each of the lines for OC1 is an average of ten five-fold cross-validation (CV) experiments; the other systems were run using a single five-fold CV design. This is because OC1 has significant amount of randomization, and the average of ten runs is a more reliable estimate than a single run.

In the table, tree size refers to the number of leaf nodes; i.e., the number of regions that the data are divided into by the tree. Tree size in a cross-validation experiment, such as this one, is the *average* size of the five trees built in the five-fold CV. Also given are the standard deviations across the five trials. The best overall accuracy, by a small margin, was produced by the tree built with CART; however, as can be seen from the standard deviation, this was not significantly better than the trees built by the other methods. Five-nearest neighbor, our baseline method, did not do as well as any of the decision trees. This could be due to the large number of features present; k-NN methods do not perform as well in large feature spaces (Salzberg 1991). In fact, we have found that feature selection methods, which reduce the number of features used, consistently improve the accuracy of

[5]By adjusting a few input parameters, the user can make OC1 build axis-parallel trees using the default goodness measures and pruning techniques of CART or C4.5. The OC1 system can thus be easily modified to mimic these other methods.

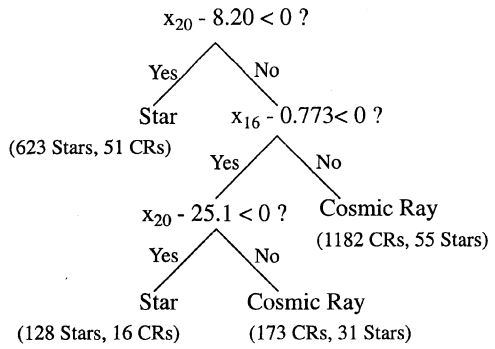

FIG. 4—Small, accurate tree produced by OC1-AP.

k-NN. Section 4.4 contains results that indicate that k-NN will in fact perform much better with fewer features. CART, OC1, and OC1-AP all produced small trees in this first experiment. The trees produced by C4.5 are larger because its pruning method is different from that of CART and OC1. It should be noted, however, that a decision tree induced on a training set having N objects can have as many as N nodes; so, considering the training set size (>1800 objects), C4.5 trees are still relatively small.

To give an example of what a particular classifier looks like, one relatively small tree that OC1-AP produced on the WF1 data is shown in Fig. 4. This tree gave 93.2% accuracy on the WF1 training set. The same tree applied to the WF2 data gave 91.9% overall accuracy.

Attribute 20 (x_{20}) is the standard deviation of the pixels in the 3×3 map. Attribute 16 (x_{16}) is the ratio of the magnitude at a radius of 2 pixels to the magnitude at 1 pixel. Because OC1 found a tree that used only two features, the partitioning induced by this tree can be viewed using those two features as the coordinate axes. In addition, it points out that it might be a good idea to run OC1 using only these two features to describe the examples, an idea which we pursue in Section 4.4.

4.2 Experiments on WF2 Data

After training each of our programs on the WF1 data, we used the same decision trees, without *any* further training, to classify the WF2 data. The purpose of these experiments was to provide an independent test of the accuracy of the methods. We used a set of images from the same region in the sky in order to equalize factors such as background noise, so that the feature values on both chips were comparable. Our results showed that there were only slight decreases in accuracy: the trees produced using WF1 data were still very good classifiers for the WF2 data. The results are shown in Table 2.

The experimental design here was very simple: we built a tree using the entire WF1 data set of 2259 objects (rather than 4/5 of the objects, as was done in the cross-validation study above), and then calculated the classification accuracy of that tree on the WF2 data set of 2368 objects. Because OC1 is randomized, we followed a slightly different procedure for it: we randomly chose 80% of the WF1 data as the training set and reserved 20% as a test set. We then built a

TABLE 2
Comparison of Decision-Tree Accuracies for WF2 Data

	Accuracy (%)			Tree
Algorithm	Overall	Stars	Cosmic Rays	Size
CART	92.5	90.0	94.6	6
C4.5	88.1	86.7	89.4	48
OC1	91.7	87.6	95.1	5
OC1-AP	91.2	87.5	94.2	8
5-NN	88.6	84.7	91.8	

decision tree with OC1 on the training set and measured its accuracy on the test set. We repeated this procedure with different random training/test partitions ten times, and chose the best (in terms of overall accuracy on its test set) of the 10 trees as the tree to use for the WF2 data. The results for that tree are the ones reported in Table 2.

Once again the CART method had a slight edge in accuracy, at just over 92%. OC1 had very slightly lower accuracy, while C4.5 and 5-NN were somewhat worse. The performance of C4.5 can be explained in part by noting that it also built a much larger tree; perhaps this tree overfit the data, resulting in the lower accuracy observed here. However, as our next experiments will show, all the methods benefit from removing additional sources of noise from the data.

4.3 Improving the Overall Accuracies

Although 92% represents a very respectable accuracy, we wanted to improve this figure. With this goal in mind, we re-examined the CCD WF1 data in an attempt to locate other sources of error. One source of noise is the so-called "hot" pixels, which occur where the "dark current" is large. The dark current is found by taking long exposures when there is no light incident on the CCD. The counts generated by this kind of exposure are caused by thermal generation (i.e., the CCD is not at 0 Kelvin) of electron-hole pairs at the interfaces between the silicon and oxide layers. At normal CCD temperature, the dark current is about 0.01 electrons/pixel s^{-1} for WF/PC. The so-called hot pixels are caused by hits from high energy CRs which damage the lattice in the bulk silicon near the CCD channel stops. Because of the high electric fields near the channel stops, the lattice damage results in the creation of electron-hole pairs at the interface between the bulk silicon and the overlying silicon oxide layer.

Some of the damaged sites in the silicon lattice anneal whenever the temperature of the CCD is raised from the cold operating temperature of -87 °C to the warm decontamination temperature of ~ 5 °C; consequently, dark frames taken months before the most recent CCD temperature cycling are not a reliable guide for finding hot pixels. For this experiment, we used an *ad hoc* approach: the hot pixels occur in all of the images in our database (10 images that are within 1 pixel of each other). We combined all 10 images using STSDAS "combine" so that we would have better signal to noise, making it easier to see the hot pixels. We used a routine called DoPhot[6] which is fairly accurate at finding cosmic rays. Since the hot pixels are usually single pixel events resembling cosmic rays, we ran DoPhot on the combined image using a threshold of 10 counts above the background, and extracted all of the objects that it identified as cosmic rays. These are presumably the hot pixels. The mean sky varied from 33 to 42 counts with an average of 39, while the standard deviation varied from 1.6 to 2.5 with an average of 2.2, giving a detection level of $\sim 4.5\sigma$. We found between 100 and 225 hot pixels this way, depending on the CCD examined. However not all of these were sufficiently intense to be detected by DAOFIND in the image from which we made the catalogue. Figure 5 shows a portion of the combined WF1 image.

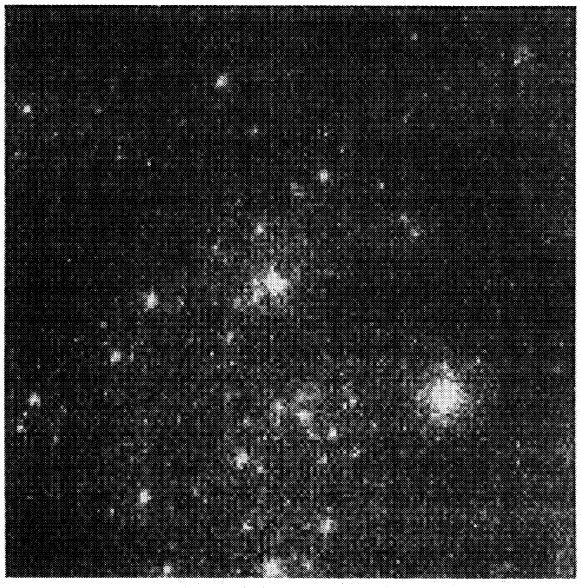

FIG. 5—A 900 s WF1 image of M81 which has been combined to improve signal to noise. This picture shows a portion of the WF1 image.

The last step, in order to find the (x,y) values of the hot pixels and remove them from the images, was to match the (x,y) coordinates of the hot pixels given by DoPhot with our star catalogue using a radius of 1.5 pixels. (The reason that we had to do this matching is that DAOFIND and DoPhot give slightly different coordinates for detections, and we needed the exact coordinates of objects to delete.)

We then removed these hot pixels from the WF1 and WF2 data sets, and re-ran all of our experiments to measure accuracy. For WF1, removing the hot pixels reduced the data set from 2259 objects to 2211, and for WF2 the number of objects dropped from 2368 to 2282.

The accuracy on both data sets improved substantially, as shown in Tables 3 and 4. As in the first experiments, the

[6]Thanks to Abhijit Saha of STScI for providing this software.

TABLE 3
Decision-Tree Accuracies for WF1 Data with Hot Pixels Removed

	Accuracy (%)			Tree
Algorithm	Overall	Stars	Cosmic Rays	Size
CART	95.8±0.9	94.2	96.6	8.4±1.5
C4.5	95.1±0.4	93.4	96.1	24.6±0.9
OC1	94.0±0.4	91.8	95.3	8.4±2.4
OC1-AP	94.9±0.4	93.6	95.6	8.8±1.7
5-NN	90.9	91.8	90.2	

WF1 accuracies were estimated using 10 five-fold cross-validation experiments. The WF2 accuracies were produced by using a single tree, created from WF1 data only, to classify the entire WF2 data set.

All of the methods except 5-NN did very well on the WF1 data. On the WF2 data, CART and OC1 did the best, while the other methods had somewhat lower accuracy. Overall, the increase in accuracy as a result of eliminating hot pixels was more than 2% for both data sets. CART improved by 3.2% and OC1 improved by 3.4% for the WF2 data.

4.4 Reducing the Feature Set

The performance of both OC1 and 5-NN can suffer when a data set is characterized by a large number of features. Because OC1 searches an exponentially large space of hyperplanes, reducing the number of attributes dramatically reduces the size of the search space, improves the efficiency of OC1's search, and results in better oblique trees. We observed earlier that some of the trees for OC1-AP and C4.5 used as few as two features: attributes 16 and 20. We therefore ran a final experiment in which we used OC1-AP, OC1, and 5-NN on the same data sets, but this time using just those two attributes. Our best results came from this final run

TABLE 4
Decision-Tree Accuracies for WF2 Data with Hot Pixels Removed

	Accuracy (%)			Tree
Algorithm	Overall	Stars	Cosmic Rays	Size
CART	95.7	96.6	95.1	8
C4.5	92.8	90.4	94.7	42
OC1	95.1	95.1	95.0	4
OC1-AP	94.0	92.5	95.1	9
5-NN	91.7	93.4	89.3	

TABLE 5
Classifier Accuracies for WF1 and WF2 Data
Using Only Attributes 16 and 20

	Accuracy (%)			Tree
Algorithm and Data Set	Overall	Stars	Cosmic Rays	Size
OC1 (WF1)	96.6	96.6	96.6	2
OC1 (WF2)	95.4	95.9	94.9	2
OC1-AP (WF1)	95.0	93.9	95.6	8
OC1-AP (WF2)	95.1	95.3	94.9	7
5-NN (WF1)	95.7	94.6	96.3	
5-NN (WF2)	95.5	95.1	95.8	

using fewer features. These results appear in Table 5. As before, the results on WF2 were produced by training on the entire WF1 data set, and then testing on WF2. In the table, we see that OC1 now produces trees that are just as accurate as those from OC1-AP and CART, and are slightly smaller. In addition, 5-NN displays a dramatic improvement and now produces a very accurate classifier. This points out the importance of selecting the right features, which in this case was done by using an axis-parallel tree method (OC1-AP) as the selection mechanism.

The tree produced by OC1 in this last experiment is displayed in Fig. 6. This figure shows the data from WF1 displayed in two dimensions, with the tree induced by OC1 superimposed. A small number of outliers have been omitted from the figure in order to improve the resolution for the rest of the data.[7] We have also not shown some objects in the middle of the dense clusters in the figure, to help the clarity.

This very small tree (just two internal nodes) in easy to understand and even easier to interpret once displayed graphically. After the fine tuning described here, a realistic estimate of accuracy for OC1 and 5-NN on the WF2 data increased from just over 92% to over 95%. We believe that this may represent the limit of accuracy for any method using the data we have available. However, larger data sets, which provide a more complete picture of the range of possible star and cosmic-ray images, might lead to additional increases.

4.5 Using Decision Trees to Confirm Labeling

In a preliminary study using paired images, OC1 produced a decision tree that discovered errors made by a standard labeling procedure used as an *HST* analysis tool. For this study, we produced two new images from the pair. The first was a CR-clipped image produced by summing the

[7]More precisely, 68 objects out of a total of 2259 objects on WF1 lie far outside the bounding box of this graph. Of these, all except six are correctly classified by the OC1 tree shown. Five of the misclassified objects are cosmic rays, and lie to the left of the oblique line. The remaining misclassified object is a star, and lies on the right of the oblique line, just above the line $X20=5.25$.

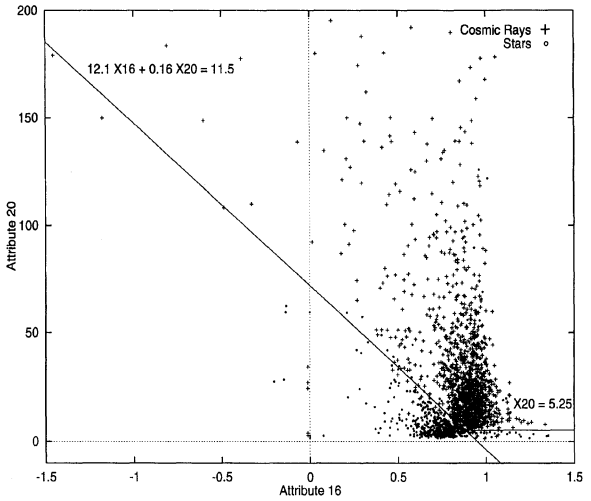

FIG. 6—Data from WF1 displayed by Attributes 16 (magnitude at a radius of 2 pixels divided by magnitude at 1 pixel) and 20 (standard deviation of pixels in 3×3 postage stamp). The OC1 tree is superimposed on the data.

original images with the STSDAS program Combine. In principle this image contains only stars. The second image was the difference of the original images, and contains only positive and negative CRs. By learning to recognize the difference in background for the stars in the summed image and the CRs in the difference image, OC1 produced a nearly perfect decision tree from a training set of 2221 objects. (The only features used for this study were the nine intensity values from the 3×3 postage stamp centered on each object.) When we classified the remaining objects, 11 objects found in the combined image were classified as CRs. Inspection of these showed that six objects were in fact CRs that had survived the Combine program because they appeared in both images; independent cosmic rays had struck the same pixels in both images, thus fooling the Combine software. In other words, our decision tree detected errors in the labeling resulting from using Combine. In retrospect, the expected number of corresponding pixels with CR hits in both images is (roughly) $1574^2/800^2 \approx 4$. After correcting the labeling in our catalogs, OC1 only mislabeled (at most) four objects: one star and three CRs, for an overall accuracy of 99.8%.

5. FUTURE WORK

Extension of Decision-Tree Classification to Aberration-Corrected Images. Our most important goal is to extend our work to the aberration-corrected images that are now being obtained with the WFPC2 that was installed in the *HST* during the 1993 December First Servicing and Repair Mission. Although the undersampling of stellar images increases by virtue of concentrating more light in a smaller number of pixels, the distinctive signature of the PSF (the sharp core surrounded by the first ring in the diffraction pattern) will be more prominent. Consequently, we are optimistic that our decision tree technique will be an excellent classifier when used on the new data.

Identification of Stars, Galaxies, Plate Flaws, and Blends in STScI Scans of the Palomar Sky Survey Plates. Our project group is currently developing classifiers for a very large database of images from the digitized version of northern and southern Schmidt telescope photographic sky surveys. The first northern survey was made in the early 1950s using the 48-in. Schmidt telescope at the Palomar Observatory in California to take pairs of red-sensitive (103a-D) and blue-sensitive (103a-O) plates of the entire northern sky; the monochromatic southern survey was made in the mid-1970s with the UK Schmidt telescope in Australia using blue/green-sensitive plates (IIIa-J). There are approximately 1500 14-in. square glass plates in the two surveys. Subsequently, the STScI commissioned a "quick V" monochromatic Palomar Schmidt survey of the northern sky using short exposures with yellow-sensitive plates (IIa-d). During the last 10 years the STScI has digitized each of the "quick V" plates and each of the northern red plates and southern blue/green plates into 14,000×14,000 pixel images.

The *HST Guide Star Catalog* was constructed from the digitized images of the "quick V" survey, but it includes only the brightest objects on the plates. There is an ongoing effort at STScI to develop the techniques required for classifying *all* the objects down to the faint limits of the deep northern red and southern blue/green plates; this problem is rather difficult, because the classifier must be able to distinguish not only stars and galaxies, but also a variety of image defects (including scratches, smudges, mold spots, dust, and finger prints!) and blends of objects.

We have already begun the process of developing a classifier by extracting a sample of 1000 objects from the northern red and southern blue/green plates. The object features were measured from the digitized images by standard STScI software, and were hand-classified by STScI experts. The experts did not examine objects that were identified as stars by a simple classifier (the "ridge" classifier, a Bayesian classifier using only the peak density and total density parameters). Objects called stars by the ridge classifier have a high probability of indeed being stars. Thus, the easy objects have already been removed from the data set. Our experience with this problem will be reported in a subsequent paper.

We thank Rogier Windhorst for a careful review of the paper and for several helpful suggestions. Thanks to Simon Kasif for numerous helpful comments. Thanks to Wray Buntine of Nasa Ames Research Center for providing the IND 2.1 package, which was the source for our versions of the C4.5 and CART systems. Thanks to Abhijit Saha of STScI for providing the DoPhot software. This work was supported in part by the National Science Foundation under Grant No. IRI-9116843.

APPENDIX: FEATURES USED FOR OBJECTS FROM CCD IMAGES

The following 20 features were computed for every object. Computation of these features is based entirely on an

object's location on the CCD and on the pixel intensities in a 3×3 grid centered on the object.

Features 1–9: Raw pixel values in a 3×3 grid.

Feature 10: x-moment (from first intensity moments).

Feature 11: y-moment (from first intensity moments).

Feature 12: Ellipicity (from a formula combining 2nd intensity moments, provided in the task IMEXAMINE in IRAF).

Feature 13: Ratio: the average of 4 pixels (above, below, left, and right of the central pixel, where the average sky value in the region has been subtracted from these intensities), divided by the intensity (minus sky) in the central pixel.

Feature 14: magnitude calculated for a radius of 1 pixel.

Feature 15: mag (radius=1.5 pixels)/mag(radius=1.0 pixels).

Feature 16: mag (radius=2.0 pixels)/mag(radius=1.0 pixels).

Feature 17: peak intensity of central pixel (radius=0.55 pixels). Computed as: total counts (in this radius)—area* (avg. sky).

Feature 18: Feature-17/mag (radius=2.0 pixels).

Feature 19: mean value of pixels in 3×3 postage stamp.Feature 20: standard deviation of pixels in 3×3 postage stamp.

REFERENCES

Breiman, L., Friedman, J., Olshen, R., and Stone, C. 1984, Classification and Regression Trees, Wadsworth International Group

Dasarathy, B. V. 1991, Nearest Neighbor (NN) Norms—NN Pattern Classification Techniques (Los Alamitos, IEEE Computer Society Press)

Freedman, W. L., et al. 1994, ApJ, 427, 628

Garey, M. R., and Johnson, D. S. 1979, Computers and Intractability—a Guide to the Theory of NP-Completeness (San Francisco, Freeman and Co.)

Heath, D. 1992, Ph.D. thesis, Johns Hopkins University

Hunter, D., Faber, S., Light, R., and Shaya, E. 1991, Final Orbital/Science Verification Report (Washington, DC, NASA)

Moret, B. M. E. 1982, Computing Surveys, 14, 593

Murthy, S. K., Kasif, S., Salzberg, S., and Beigel, R. 1993, Proc. of the Eleventh National Conference on Artificial Intelligence, (Washington, MIT Press), p. 322

Murthy, S. K., Kasif, S., and Salzberg, S. 1994, Journal of Artificial Intelligence Research, 2, 1

Quinlan, J. R. 1993, C4.5: Programs for Machine Learning (San Mateo, Morgan Kaufmann)

Salzberg, S. 1991, Methodologies for Intelligent Systems, 6th International Symposium, p. 399

Windhorst, R. A., Franklin, B. E., and Neuschaefer, L. W. 1994, PASP, 106, 798