PERGAMON

Pattern Recognition 34 (2001) 1247–1256

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

# Locating geometric primitives by pruning the parameter space

Clark F. Olson*[,1]

*Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, MS 125-209, Pasadena, CA 91109-8099, USA*

Received 27 July 1999; received in revised form 27 March 2000; accepted 27 March 2000

**Abstract**

This paper examines the detection of geometric primitives using 2D edge pixels or 3D range points. The geometric primitives are described by parametric equations in the image space. An explicit error model allows the primitives to be extracted robustly, while a hierarchical search of the parameter space with conservative pruning allows the primitives to be located efficiently. The result is an efficient search strategy that is robust to distractors, missing data and noise, and that does not require an initial estimate of the positions of the geometric primitives. We apply these techniques to circle detection, for locating craters on planetary bodies and analyzing engineering drawings, and to cylinder detection, for finding unexploded ordnance in test ranges. © 2001 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

*Keywords:* Geometric primitive extraction; Parameter estimation; Divide-and-conquer search; Robust error modeling; Randomized algorithms

## 1. Introduction

The extraction of geometric primitives from image data is a useful tool in many applications. In some applications, such as transcribing engineering drawings or crater detection, the extraction of geometric primitives is an end result. Other applications use the extracted primitives for higher-level analysis and recognition techniques. In either case, applications in which geometric primitive extraction is necessary require robust, accurate, and efficient results. This paper presents a new method for extracting geometric primitives that has these characteristics.

Several methods have been previously used to extract geometric primitives from image data. The most popular methods are variations on the Hough transform (see Refs. [1,2] for reviews). These methods map the image features (such as edge pixels in an image or surface points in a range map) into the space of possible primitive parameters and then search for peaks in this parameter space, since these peaks correspond to likely geometric primitives. Other approaches include the use of robust statistics to fit the image data in the presence of noise and outliers [3,4], methods that hypothesize primitive positions using small sets of data features and then test each position that is hypothesized [5,6], minimization of a cost function through iterative optimization [7], and region growing [8].

A key drawback to many of these methods is that it is difficult to both propagate the effects of localization error in the data features and handle large amounts of distracting data (a particular primitive may consist of a small fraction of the total data). Methods that are robust tend to be inefficient. In this work, we take an approach that deals with the effects of localization error explicitly through the use of a bounded error model, can handle large amounts of distracting data, does not require an initial estimate of the primitive positions, and is computationally efficient.

Our approach is inspired by research on object recognition in which the parameter space is recursively divided and pruned [9,10]. We retain the splitting and pruning search strategy, but rather than using discrete points as our model, we use a parameterized equation to describe the geometric primitives. Similar ideas are also used in

* Tel.: + 1-818-354-0638; fax: + 1-818-393-4085.
*E-mail address:* olson@robotics.jpl.nasa.gov (C.F. Olson).
[1] http://robotics.jpl.nasa.gov/people/olson/homepage.html

the fast Hough transform [11], although this method applies only to linear models and lacks robustness since it does not propagate error into the parameter space.

We extract geometric primitives from the image data by searching for parameters corresponding to primitives that satisfy an acceptance criterion based on how many of the data features are fit by the primitive up to a bounded error. In order to perform this search, we consider cells of the parameter space. For each cell that is examined, we determine whether the cell could contain the parameters for a primitive that meets the acceptance criterion. If not, then the cell is pruned. Otherwise, the cell is split into two subcells and the subcells are examined recursively. When a very small cell is reached, we test the primitive at the center of the cell to determine if it meets the acceptance criterion.

An interesting facet of our search strategy is that a hierarchy is constructed not only in the parameter space, but also in the image feature space. This allows many image features to be pruned at each step with little computation, in addition to the pruning in the parameter space. Empirical evidence suggests that this technique reduces the complexity of the extraction process. This use of a feature hierarchy is a general technique that can also be applied to previous methods that use a search strategy that prunes the parameter space [9–11]. Robust random sampling techniques are also used to improve the speed of the search.

## 2. Search strategy

In order to describe the search strategy that we use, we must first discuss how the primitives will be represented and how we decide which primitives should be reported.

Each class of primitives is represented by a set of parametric equations $f(X;\Gamma) = 0$, where $X$ is a vector of the data feature parameters and $\Gamma$ is a vector of the primitive parameters. For example, circles may be represented with $X = [x\ y]^t$ and $\Gamma = [x_c\ y_c\ r]^t$ by the following equation:

$$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0.$$

We use an acceptance criterion based on the number of data features that lie within some allowable error $e$ of the primitive. We denote this number $N_e(\Gamma)$, where $\Gamma$ is some primitive position. When primitives of arbitrary scale are considered, it is often useful to normalize this count by the scale of the primitive (e.g. for circles, we would divide by the radius), so that there is no bias towards large primitives. Either the best primitive according to the criterion can be reported, or all primitives that surpass some threshold $T$. This criterion is able to handle occlusion by setting the threshold at an appropriate value. While lower thresholds can result in false positive instances, this is a universal problem for primitive extraction that must be dealt with if we wish to also detect occluded primitives.

To find the primitives that satisfy the acceptance criterion, we consider rectilinear cells in the parameter space. (At the start of the search, the parameter space may consist of a single large cell or it may be split into subcells.) Each point in the parameter space represents a possible position of a geometric primitive in the data. The cells are volumes of the parameter space and thus represent a continuous space of possible geometric primitive locations. Each cell in the parameter space is tested to determine if it can contain the parameters of a primitive that satisfies the acceptance criterion. An efficient testing mechanism is used that is conservative in that it never rules out a cell that contains a good primitive, but it can fail to rule out a cell that does not contain any good primitive. This does not result in false positives since the cells are recursively examined at finer resolutions until we reach very small cells. The smallest cells are tested by considering the primitive represented by the center of the cell.

In order to test each cell (aside from the smallest cells), we compute a bound on the number of image features that can match any primitive in the cell. To accomplish this, we consider the primitive $\Gamma_C$ represented by the center of the cell. For each image feature $p$, we sum the distance $d_p$ from the feature to this primitive and the maximum deviation in the distance that can be achieved by any other primitive in the cell. In general, this deviation is a function of the size and position of the cell and the position of the image feature:

$$d_{C,p} = \max_{\Gamma \in C} \|d(f(X;\Gamma),p) - d(f(X;\Gamma_C),p)\|,$$

where $d(f(X;\Gamma),p)$ is the distance from the primitive represented by $f(X;\Gamma)$ to the image feature, $p$.

For the sake of efficiency, we place a bound on this value for any feature in the image:

$$d_C = \max_{\Gamma \in C} \max_{p \in D} \|d(f(X;\Gamma),p) - d(f(X;\Gamma_C),p)\|,$$

where $D$ is the set of image features (see below for an example).

We next count the number of image features such that the bound on the distance from any primitive in the cell, $d_p - d_C$, is less than the allowable error $e$. The cell can be pruned if this count is below the threshold set by the acceptance criterion, since we can guarantee that the cell cannot contain a primitive that meets the acceptance criterion. If the cell cannot be pruned, it is divided into two subcells by slicing it at the midpoint of one of the parameters and the subcells are considered recursively using a depth-first search.

To prevent this method from dividing the cells until they become arbitrarily small, we set a threshold on $d_C$.

When $d_C$ falls below the threshold, we test the cell by considering the quality of primitive at the center of the cell to see if it meets the acceptance criterion. This is equivalent to imposing some underlying discretization on the parameter space, and considering only those positions in the underlying discretization. It is possible that this could cause a primitive that meets the acceptance criterion to be missed, but since the cells are very small when they are tested in this manner, it is unlikely that a significant error will be made. To formalize this, it can be shown that, when $d_C < \delta$, we are guaranteed to find any primitive for which $N_{e-\delta} > T$.

The computation of $d_C$ varies depending on the class of primitives that we are trying to locate. We examine the detection of circles as an example in this section. When we wish to detect circles, we use $f(X,\Gamma) = (x - x_c)^2 + (y - y_c)^2 - r^2$. Given two circles, $\Gamma_1 = [x_{c_1}, y_{c_1}, r_1]^t$ and $\Gamma_2 = [x_{c_2}, y_{c_2}, r_2]^t$, the difference in the distance of any point from these two circles can be bounded by distance between the centers of the circles and the difference in radii.

$$d \leqslant \sqrt{(x_{c_1} - x_{c_2})^2 + (y_{c_1} - y_{c_2})^2} + |r_1 - r_2|.$$

Note that this does not depend on the circles themselves, only the difference between the parameters, $\delta = \Gamma_1 - \Gamma_2$, so we can rewrite this using $\delta$,

$$d \leqslant \sqrt{\delta_x^2 + \delta_y^2} + |\delta_r|.$$

Now, for any cell in the parameter space,

$$C = \{[x_c, y_c, r]^t \mid x_l \leqslant x_c \leqslant x_h, y_l \leqslant y_c \leqslant y_h, r_l \leqslant r \leqslant r_h\},$$

we can use this relationship to place a bound on $d_C$:

$$d_C \leqslant \sqrt{\left(\frac{x_h - x_l}{2}\right)^2 + \left(\frac{y_h - y_l}{2}\right)^2} + \left(\frac{r_h - r_l}{2}\right).$$

This bound on $d_C$ will not always allow us to prune the cell from consideration. When a cell cannot be pruned, we must decide which parameter should be chosen to subdivide the cell. We subdivide the parameter that causes our bound on $d_C$ to be reduced the most.

## 3. Image feature hierarchy

We can improve the search strategy described above by performing hierarchical pruning not only in the parameter space, but also in the image feature space. It is possible to rule out a cell containing multiple image features for a particular parameter space cell by examining a single image location using a pruning technique that generalizes the basic search strategy.

Consider a cell $I$ in the image feature space. Let $p_I$ be the center of this cell, $d_I$ be the distance from $p_I$ to the furthest image feature in the cell, and $d_{p_I}$ be the distance from $p_I$ to the primitive represented by $\Gamma_C$ (the center of the parameter space cell $C$). We redefine $d_C$ such that the bound must now apply only to the image features in $I$:

$$d_C = \max_{\Gamma \in C} \max_{p \in I} \|d(f(X;\Gamma), p) - d(f(X;\Gamma_C), p)\|.$$

If $d_{p_I} - d_I - d_C > e$, then no image feature in the image cell can match any primitive in the parameter space cell up to the error, $e$, and we can thus prune the image cell for this parameter space cell and any of its subcells that are examined.

To take advantage of this idea, we build a hierarchy of image feature cells that is similar to an R-tree [12], see Fig. 1. This is a binary tree in which each node corresponds to a cell in the image feature space. The root of the tree is a cell just large enough to contain all of the image features. Each cell that contains more than one image feature has two children that are roughly half its size. The union of the subcells need not be equivalent to the parent cell, though, since we need only ensure that the subcells cover all of the image features in the original cell. Each individual feature is a leaf of the tree. This tree is built recursively by subdividing the cells at the midpoint of the longest axial direction. Each subcell is then contracted such that it is just large enough to contain all of the data features within it.

Once we have built this data structure, we can improve the efficiency of the search strategy by pruning cells in the image hierarchy, in addition to the pruning performed in the parameter space. When a particular parameter space cell is examined, we search this tree of cells in the image space, rather than considering each feature individually.

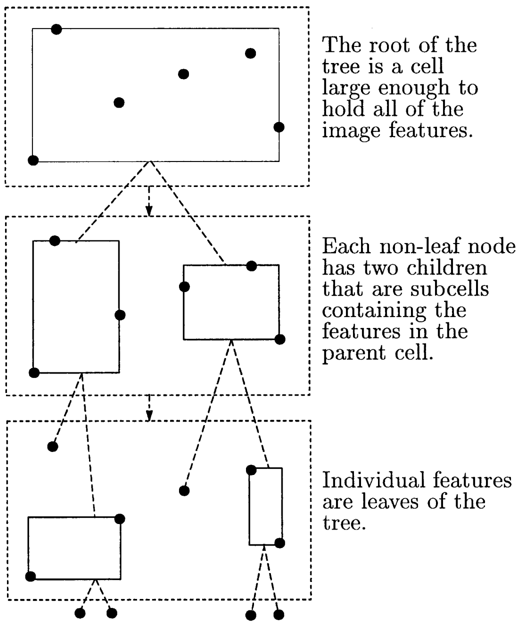

The root of the tree is a cell large enough to hold all of the image features.
Each non-leaf node has two children that are subcells containing the features in the parent cell.
Individual features are leaves of the tree.

Fig. 1. The image features are recursively subdivided into cells.

This is performed in a manner similar to the search over the parameter space cells. For each image space cell that is examined, we determine if it can be pruned. If the cell cannot be pruned, then we examine its subcells, unless it is a leaf, in which case we increment the count on the number of features that cannot be pruned.

Note that any image feature cell that is pruned for some parameter space cell need not be considered in the recursive examination of the parameter space subcells, since the primitives represented by the subcells are a subset of those represented by the original cell.

This technique is general enough to be used in combination with previous object recognition and curve detection methods (e.g. Refs. [9–11]), and the performance of these methods can be improved through its use.

## 4. Random sampling

In many cases, there is more data than necessary to extract a geometric primitive. For example, a cylinder that covers only a $25 \times 100$ patch in a range image yields 2500 points on the cylinder, which is far more than is necessary to extract the cylinder. We can use random sampling techniques to reduce the amount of data that is examined, while only slightly decreasing the accuracy of the techniques. To accomplish this, we randomly sample some set of the image features, which are processed as described above. If we find a cylinder that matches enough of the data features then we test the cylinder using all of the data features.

We assume that only geometric primitives consisting of at least some fraction $\alpha$ of the $n$ image features are salient and must be located. If we sample $s$ features and test the full cylinder if $\gamma s$ of the sampled features belong to a primitive then the probability that a primitive is missed due to sampling is

$$P_{\text{fail}} = \sum_{k=0}^{g-1} \frac{\binom{\alpha n}{k}\binom{n-\alpha n}{s-k}}{\binom{n}{s}},$$

where $g = \lceil \gamma s \rceil$ and $n$ is the number of image features. We compute $\gamma$ numerically such that a small predetermined error rate (e.g. $P_{\text{fail}} < 0.01$) is satisfied. For example, a $256 \times 256$ range image containing a cylinder composed of 2500 pixels has $\alpha = 0.0381$. If 1000 points are sampled and a probability of failure no more than 0.01 is desired, then we can use $\gamma \leqslant 0.0260$ to achieve this.

## 5. Overall algorithm

The complete algorithm using the steps in the above sections is given in Fig. 2. We first subsample the set of data points (line 3), if necessary. The data is then pre-processed to obtain the data structure described in Section 3, which is performed with a function call to *pre-process* on line 15 of the pseudo-code. This preprocessing step uses a recursive method, where the set of points is divided into two subsets and then each of the subsets is processed. The recursion ends when we examine individual points of the set (lines 16–17). Otherwise, this procedure builds up a binary tree storing the set of data points in a hierarchy determined by the geometry of the set (lines 19–24).

Next, we examine the pose space using a recursive procedure, starting with a cell that contains the entire space. For each cell that is examined, we use the *process-cell* procedure (line 25) to count the number of data points that are consistent with some primitive in the pose space cell by traversing the tree of pre-processed data points, pruning branches when possible using the test described in Sections 2 and 3 (lines 30 and 37), in which case a value of zero is returned (lines 33 and 40). When the cell is not pruned we return the sum of the values from the childen of the cell (line 38), unless the cell is a leaf, in which case we return 1 (line 31).

Finally, we examine the number of consistent points reported by *process-cell* (line 6). If the count is above our threshold (which may be a function of the cell position to take into account information such as the radius of the primitive), then we either report a candidate location (if the pose space cell is a leaf) or divide the cell into subcells to examine recursively (lines 10–12).

## 6. Performance

The empirical performance of this method for extracting geometric primitives has been tested in several experiments. We note that the computation complexity of these techniques is $O(pn)$, where $p$ is the number of positions in the underlying discretization of the parameter space and $n$ is the number of data features present in the image. We can achieve this complexity with a brute force search of every position in the discretization of the parameter space and checking whether each feature is within $e$ of the geometric primitive at that position. The pruning techniques do not improve upon the worst-case complexity, since we cannot guarantee that the number of positions in the parameter space that are examined is less than $O(n)$. However, this complexity is not informative, since the pruning and randomization techniques make a pronounced difference in the running time, and, indeed, reduce the empirical complexity observed in real images.

We have performed experiments for circle detection in two-dimensional data to examine the empirical running time of the techniques with respect to several parameters. Random sampling was not used in these experiments, in order to test the performance of the pruning techniques.

```
locate($D, C, T, e$):           /* $D$ is the set of data points. $C$ is the current pose cell. */
                                /* $T$ is the acceptance function and $e$ is the allowable error */
    Subsample $D$ (if necessary).
    $I$=pre-process($D$).
    Count=process-cell($I, C, e$).
    If (Count$\geq T(C)$) then
        If ($C$ is a leaf) then
            Report a candidate location at $C$.
        Else
            Bisect $C$ along longest dimension into $C_{begin}$ and $C_{end}$.
            locate($D, C_{begin}, T$)
            locate($D, C_{end}, T$)
    Else
        Prune $C$.

pre-process($D$):                          /* $D$ is the current set of data points. */
    If ($\|D\| = 1$) then
        Return D
    Else
        Select the longest dimension of $D$.
        Bisect longest dimension at midpoint $p$.
        Divide the points in $D$ into two sets: $D_{begin}$ and $D_{end}$.
        Struct->First=pre-process($D_{begin}$)
        Struct->Last=pre-process($D_{end}$)
        Return Struct

process-cell($I, C, e$):                 /* $I$ is a set of pre-processed data points. */
                          /* $C$ is the current pose cell and $e$ is the allowable error. */
    Compute $c$ (the center of $C$).
    Compute $d_C$ (this is problem dependent).
    If ($I$ is a leaf) then
        If (dist$(I, c) - d_C \leq e$) then
            Return 1
        Else
            Return 0
    Else
        Compute $i$ (the center of $I$).
        Compute $d_I$ (the radius of points in $I$).
        If (dist$(i, c) - d_C - d_I \leq e$) then
            Return process-cell(I->First),$C, e$)+process-cell(I->Last),$C, e$)
        Else
            Return 0
```

Fig. 2. Pseudo-code of the algorithm.

We first tested the change in the running time as a function of the number of features in the image. This experiment was performed with images of varying size, but with the same feature density, thus images with more features correspond to larger images. Fig. 3 shows the results. It can be observed that the running time of the techniques is very closely linear up to 10,000 image features.

If we instead vary the density of the pixels in an image of the same size, the results are very different, see Fig. 4. In this case, we held the threshold at which circles were reported constant, so that in the lower density images, no circles were found. The running time appears to be exponential in the density of the image. This, in part, shows why the asymptotic complexity mentioned earlier is misleading. The amount of work that is done is still bounded by a linear function of the number of features and the number of possible model positions. However, due to the optimizations, we examine far fewer than the total number of possible model positions and, for each model position, we do not examine all of image features. The amount of work that is done with the optimizations rises sharply with the density of the features in the image.

These two graphs indicate that choosing an appropriate threshold for the given density of image features is crucial to maintaining high efficiency. If a threshold is chosen such that the number of primitives meeting the acceptance criterion due to the random accumulation of features is large, then the running time of the detection techniques is likely to be poor. However, when the threshold is chosen such that a small number of significant primitives are detected, the running time will be good.

In order to determine the benefit gained by the pruning techniques, we have tested a method that does not take advantage of the image space pruning. Fig. 5 shows the speedup that is achieved by the pruning in the image space through the use of our image feature hierarchy over

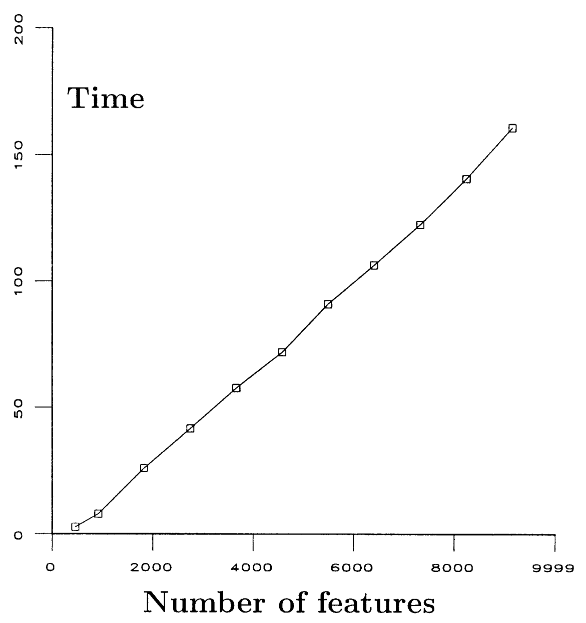


Fig. 3. When the threshold on which circles are detected is varied with the density of the image features, the running time of the techniques grows slowly.

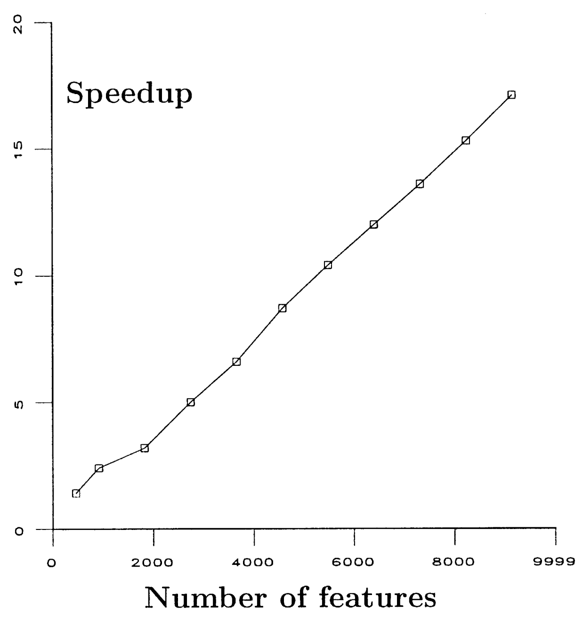


Fig. 5. The speedup that is achieved through the using of pruning in the image space is linear in the number of image features.

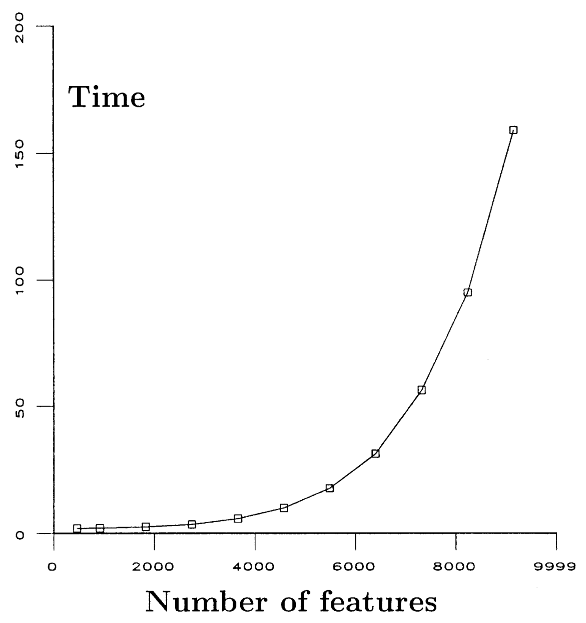


Fig. 4. The running time of the techniques appears to be exponential when the density of the image features is increased without changing the threshold on which circles are detected. However, maintaining the same threshold as the density increases significantly is not realistic in practice.

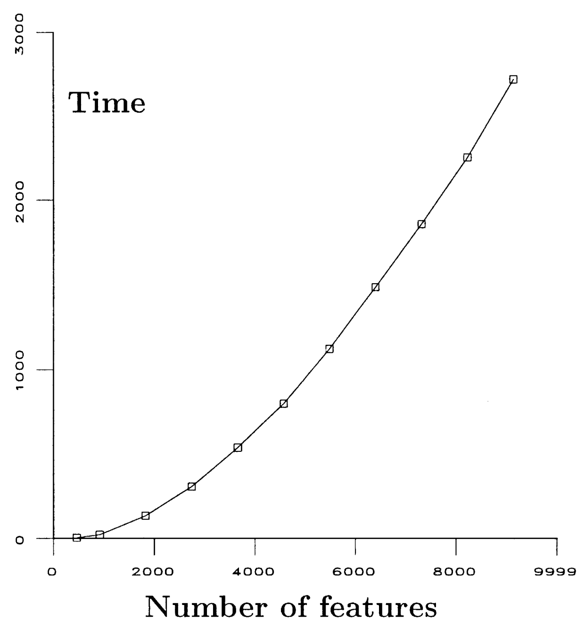


Fig. 6. The time required when the image space pruning techniques are not applied is quadratic in the number of image features.

a method that does not use image space pruning. In these experiments, the number of data features was increased through the examination of various size images with the same image density. The speedup that is achieved through the use of the image feature hierarchy increases with the size of the image, as expected. Since this speedup is linear, and the running time also increases approximately linearly with the number of features, the running time of the techniques without this pruning should be approximately quadratic in the number of features. This is supported by the running times observed, see Fig. 6. It thus appears that the use of these pruning techniques yields a decrease in the practical complexity of the primitive extraction algorithm from quadratic to linear in the size of the image.

The observed quadratic complexity for the case where image pruning is not used is because there is not only a linear increase in the possible positions at which the model may appear, but also a linear increase in the number of features processed at each of the positions that are examined. The pruning techniques effectively eliminate the linear factor due to the number of features processed, thus reducing the overall computation time to linear in the size of the image.

## 7. Relationship to previous work

The techniques we have described can be thought of as an efficient implementation of robust Hough transform techniques. The reporting criterion that we use is the same as in a robust version of the Hough transform [13] and the method detects the geometric primitives by searching the parameter space. However, the method by which the space is searched is somewhat different from a conventional Hough transform implementation. Instead of explicitly mapping sets of image features into a quantized parameter space, the parameter space is considered as a recursive hierarchy of cells, each of which is tested to determine whether it could contain a primitive satisfying the acceptance criterion.

The fast Hough transform technique of Li et al. [11] uses a strategy that is similar to ours to search the parameter space, since the parameter space is recursively divided and pruned. However, their technique is limited to linear problems, such as line or plane detection. In addition, they use an acceptance criterion that is not robust. For each cell in the parameter space, they count only the image points that fall exactly on a primitive that is represented by a point in the cell. When the cells become small, the error in the image feature detection will cause points to fall outside of the cell that represents the best primitive. This criterion relies upon the smallest cells being large enough to catch most of the points from the correct primitives, while being small enough not to detect false positives. However, this technique is prone to failures [13,14].

The most closely related work to ours is Breuel's method for line detection under bounded error [15]. Breuel uses both a similar search strategy and a robust acceptance criterion. This paper is limited to the detection of lines, for which Breuel gives a very detailed analysis, while we consider techniques that apply to any parameterized geometric primitive. Our work can thus be considered to be a generalization of Breuel's work to arbitrary geometric primitives with the additional extension of image space pruning.

We note that it is possible to use object recognition techniques directly in the extraction of geometric primitives, by constructing a canonical primitive composed of a set of discrete points [10]. However, there are two disadvantages to this approach. First, if we do not know the scale of the primitive, the sampling of points on the primitive will be either too coarse, and thus a poor representation, or too fine, and thus inefficient in the search. Second, some primitives (such as cylinders) have unbounded potential extent. If we place an artificial bound on the extent of such primitives, the extent is likely to be too short or too long, leading to problems as above. In addition, this adds a degree of freedom to the search, since we must now search the translations along the axis of the cylinder. This adds considerably to the computation required by the method.

## 8. Results

We have applied these techniques to two problems. In the first, we perform circle detection in order to extract craters from imagery of planetary bodies and to analyze engineering drawings. The second considers locating surface-lying ordnance by extracting cylinders from three-dimensional range data. Of course, there are many other applications for extraction of circles and cylinders from image data and also many other primitives that can be extracted through the use of these techniques.

### *8.1. Circle detection*

For several applications, it is desirable to be able to detect craters on planetary bodies. Some examples include mapping of planetary bodies, geological studies, and optical navigation of spacecraft. One approach to accomplishing this is to perform edge detection on an image of the surface of a planetary body and detect the circles (or ellipses) that are present in the edge image. We use the following parameterization for circles:

$$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0.$$

In this application it is useful to incorporate knowledge about the lighting direction in the extraction process, since the gradient orientations at the edges of craters will have roughly the same orientation as the lighting direction (positive dot-product), while shadow edges will have the opposite orientation (negative dot-product). We can thus screen out the shadow edges through the use of this information.

Fig. 7 shows an application of these techniques to detecting craters on the surface of Phobos (a moon of Mars). Edges were first detected in an image from *Viking*

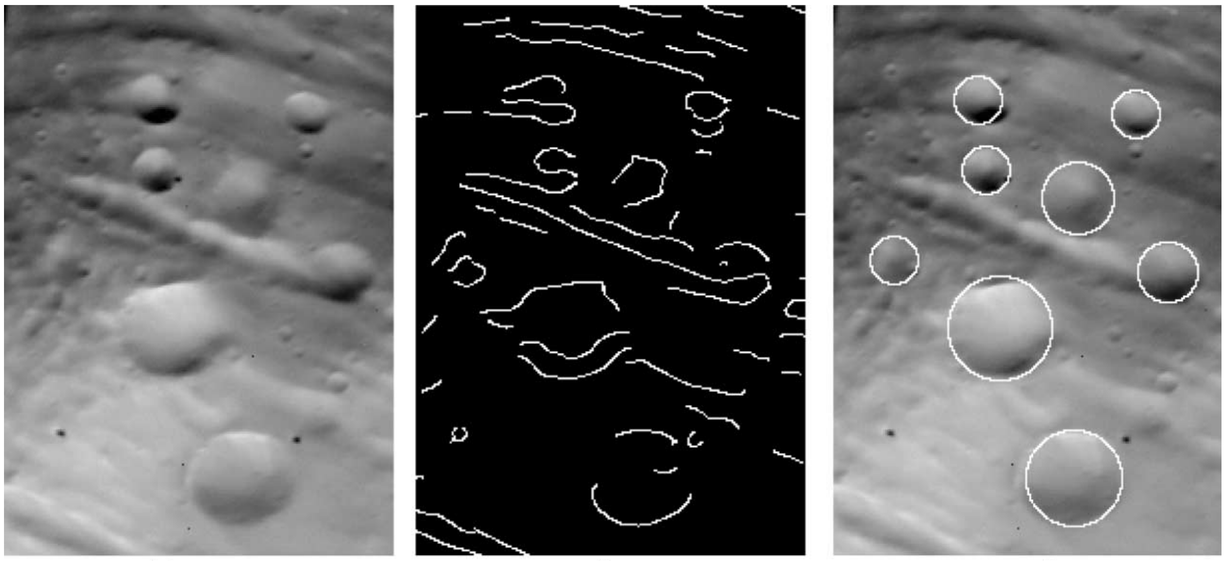
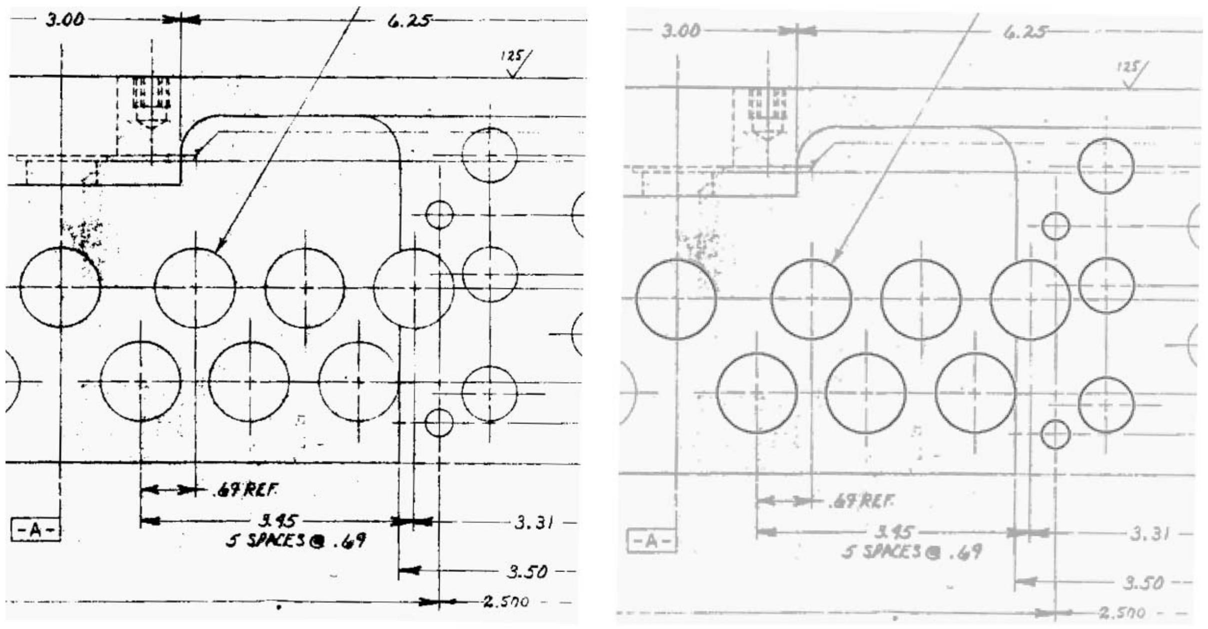

(a) (b) (c)

Fig. 7. Finding craters by locating circles in edge images: (a) an image of the surface of Phobos (the larger of the two moons of Mars), (b) the edges detected in the image, (c) the craters detected overlaid on the original image.



(a) (b)

Fig. 8. Circle detection example in a noisy, scanned engineering drawing: (a) original image, (b) circles detected.

*Orbiter*. We then searched for the circles in the edge map that had $N_{1.5}(\Gamma)/r \geqslant \pi$, where $N_{1.5}(\Gamma)$ is the number of image pixels that matched the circle up to a maximum error of 1.5 pixels and $r$ is the radius of the circle. We did not use the random sampling techniques for this case, since the volume of the data was not large. Eight circles were found that met the acceptance criterion and these correspond to the eight significant craters in the image.

Fig. 8 shows an example where these techniques have been applied to a noisy, scanned engineering drawing. In this case, we searched for circles in the edge map where $N_{1.0}(\Gamma)/r \geqslant 2\pi$, where $N_{1.0}(\Gamma)$ is the number of image

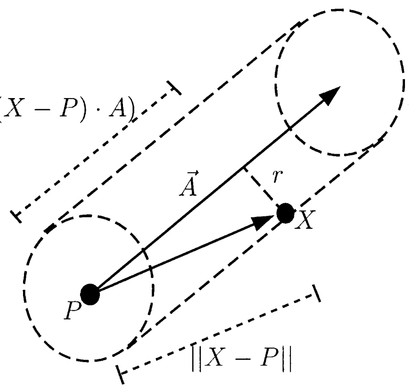


Fig. 9. Parameterization of a cylinder.

pixels that matched the circle up to a maximum error of 1.0 pixels and $r$ is the radius of the circle. Each of the complete circles was detected in the image and the significant clutter in the image did not result in any false positives being found.

### 8.2. Cylinder detection

A second problem that we have examined is the localization of unexploded, surface-lying ordnance using stereo imagery for the purpose of semi-autonomous remediation. This is motivated by the existence of many test ranges in the United States (and abroad) where live-fire testing is performed that contain dangerous ordnance. In current practice, technicians walk the range marking the locations of ordnance for disposal, a dangerous task. We use a binocular stereo system [16] to generate range images of outdoor scenes. We extract cylinders from stereo range data to determine if there is an unexploded bomb present.

Cylinders have five degrees of freedom, but in this application, we assume that the radius of the bomb is known so that the search space is reduced to four parameters. We also use the known geometry of cylinders to prevent portions of the cylinder that should be self-occluded from matching points in the image. We parameterize the cylinders with the equation

$$||X - P||^2 - ((X - P)A)^2 - r^2 = 0,$$

where $X$ describes the points on the cylinder, $P$ is an arbitrary point on the cylinder axis, $A$ is the axis direction, and $r$ is the radius of the cylinder, see Fig. 9. This parameterization allows multiple representations of the same cylinder, since it has seven free parameters rather than five. We incorporate two constraints so that we have a unique representation for each possible cylinder:

$$||A|| = 1$$

and

$$A \cdot P = 0.$$

Fig. 10(a) shows an example image containing a bomb where we have applied the cylinder detection techniques. Fig. 10(b) shows the surface map that was extracted using stereo vision. In this case, we sampled 5% of the range data and used numerical techniques to determine an appropriate threshold for testing to the entire data set. A cylinder was detected at the location of the bomb shown in Fig. 10(c). No other cylinders were detected.

## 9. Summary

This paper has explored the detection of geometric primitives in image data. We have adopted a search strategy where the parameter space is recursively divided

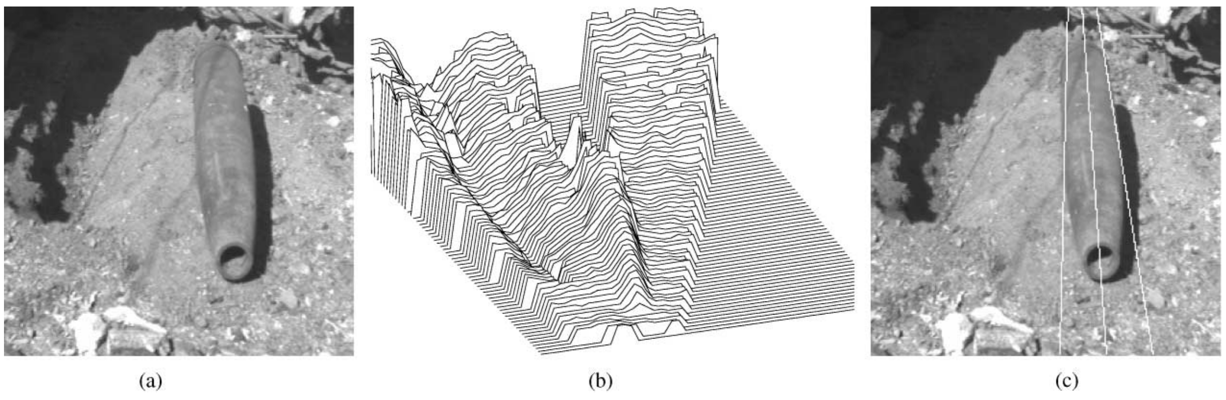


Fig. 10. The use of cylinder detection to find unexploded ordnance. (a) Left image of a stereo pair containing surface-lying ordnance. (b) Range map extracted from the stereo pair. (c) A cylinder meeting the acceptance criterion was found at the location of the bomb. This image shows the axis and boundaries of the cylinder.

and pruned. This allows geometric primitives to be robustly and efficiently extracted from noisy data that contains large amounts of distracting data, without requiring an initial estimate of the primitive locations.

Further improvements have been gained through the use of a hierarchical representation of the image features that allows them to be efficiently processed and the use of random sampling to reduce the volume of data that must be processed. The use of this hierarchical image representation to reduce the search time is a general technique that can be applied in addition to previous algorithms. Empirical experiments indicate that this technique reduces the search time for quadratic to linear in the number of image features.

Finally, these techniques have been applied to the detection of craters on planetary bodies, analysis of engineering drawings, and locating surface-lying ordnance in military test ranges.

## Acknowledgements

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Wright Laboratory at Tyndall Air Force Base, Panama City, Florida, through an agreement with the National Aeronautics and Space Administration.

## References

[1] J. Illingworth, J. Kittler, A survey of the Hough transform, Comput. Vision Graphics Image Process. 44 (1988) 87–116.

[2] V.F. Leavers, Which Hough transform? CVGIP: Image Understanding 58 (2) (1993) 250–264.

[3] D.S. Chen, A data-driven intermediate level feature extraction algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 11 (7) (1989) 749–758.

[4] P. Meer, D. Mintz, A. Rosenfeld, D.Y. Kim, Robust regression methods for computer vision: a review, Int. J. Comput. Vision 6 (1) (1991) 59–70.

[5] R.C. Bolles, M.S. Fischler, A RANSAC-based approach to model fitting and its application to finding cylinders in range data, Proceedings of the International Joint Conference on Artificial Intelligence, 1981, pp. 637–642.

[6] G. Roth, M.D. Levine, Extracting geometric primitives, CVGIP: Image Understanding 58 (1) (1993) 1–22.

[7] F. Solina, R. Bajcsy, Recovery of parametric models from range images: the case for superquadrics with global deformations, IEEE Trans. Pattern Anal. Mach. Intell. 12 (2) (1990) 131–147.

[8] P.J. Besl, R.C. Jain, Segmentation through variable-order surface fitting, IEEE Trans. Pattern Anal. Mach. Intell. 10 (2) (1988) 167–192.

[9] T.M. Breuel, Fast recognition using adaptive subdivisions of transformation space, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1992, pp. 445–451.

[10] D.P. Huttenlocher, W.J. Rucklidge, A multi-resolution technique for comparing images using the Hausdorff distance, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1993, pp. 705–706.

[11] H. Li, M.A. Lavin, R.J. Le Master, Fast Hough transform: a hierarchical approach, Comput. Vision Graphics Image Process. 36 (1986) 139–161.

[12] A. Guttman, R-trees: a dynamic index structure for spatial searching, Proceedings of the ACM SIGMOD International Conference on Management of Data, 1984, pp. 47–57.

[13] C.F. Olson, Constrained Hough transforms for curve detection, Comput. Vision Image Understanding 73 (3) (1999) 329–345.

[14] W.E.L. Grimson, D.P. Huttenlocher, On the sensitivity of the Hough transform for object recognition, IEEE Trans. Pattern Anal. Mach. Intell. 12 (3) (1990) 255–274.

[15] T.M. Breuel, Finding lines under bounded error, Pattern Recognition 29 (1) (1996) 167–178.

[16] L. Matthies, Stereo vision for planetary rovers: stochastic modeling to near real-time implementation, Int. J. Comput. Vision 8 (1) (1992) 71–91.

**About the Author**—CLARK F. OLSON received the B.S. degree in Computer Engineering and the M.S. degree in Electrical Engineering from the University of Washington, Seattle in 1989 and 1990, respectively. He received the Ph.D. degree in Computer Science from the University of California, Berkeley in 1994. After spending two years as a post-doctoral research at Cornell University, he was employed by the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, where he is currently a member of the Machine Vision group. His research interests include computer vision and mobile robotics.