Genetically Enhanced Feature Selection of Discriminative Planetary Crater Image Features

Joseph Paul Cohen, Siyi Liu, and Wei Ding

Department of Computer Science
The University of Massachusetts Boston
100 Morrissey Blvd. - Boston, MA 02125-3393
{joecohen,silu,ding}@cs.umb.edu

Abstract. Using gray-scale texture features has recently become a new trend in supervised machine learning crater detection algorithms. To provide better classification of craters in planetary images, feature subset selection is used to reduce irrelevant and redundant features. Feature selection is known to be NP-hard. To provide an efficient suboptimal solution, three genetic algorithms are proposed to use greedy selection, weighted random selection, and simulated annealing to distinguish discriminate features from indiscriminate features. A significant increase in the classification ability of a Bayesian classifier in crater detection using image texture features.

Keywords: machine learning, genetic algorithms, crater detection, bayesian classifier

1 Introduction

Impact craters are structures formed by collisions of meteoroids with the planetary surface. The importance of impact craters stems from the wealth of information that detailed analysis of their distributions and morphology can bring forth. Crater counting is the only technique for establishing relative chronology of different planetary surfaces. However, crater detection from planetary images is a difficult problem because of the complex geological surface structure of remote planets. If an acceptable solution is found it will enable many studies including determining the geologically active regions of a planet, relatively dating sections of a planet, and determining both landing and exploration sites for interplanetary robots. The challenge currently is to achieve an acceptable level of accuracy as required by planetary domain scientists [11] [5] [10] [6] [12].

The state of the art method of crater detection involves utilizing the texture and contrast of the crater image [3]. This is achieved by extracting numerical features from an image, each representing a particular texture or contrast, and then applying machine learning to decide if potential crater images are in fact craters. Haar features, a gray-scale image texture features, are especially useful because of their ability to be calculated efficiently using a data structure called integral images [13].

The challenge in using Haar features is that the number of Haar features can easily be tens of thousands. Many Haar features are redundant or even irrelevant. The curse of dimensionality is inevitable if we do not select subsets of features that are useful to build a good classifier. All features generated from the image can be broken down into discriminate features and indiscriminate features. Discriminate features contain information that is useful during classification. Indiscriminate features provide no information to the classifier or misguiding information.

The goal of feature selection is to select the optimal subset of features for some classifier. Feature subset selection is known to be NP-hard. Exhaustive search is the only way to find the optimal subset of a set of features. To find, for certain, the optimal solution all permutations must be considered. The search space is 2^f where f is the number of features. For an example with only 58 features it would take 91,336,645.5 years to compute all classifiers if a classifier took 0.10 seconds to create and evaluate.

Three algorithms are presented in this paper, using approaches of highest fitness selection, weighted random selection, and simulated annealing, to select discriminate features that a classifier will use to classify images. These algorithms are given a set of features and return a suboptimal subset. The algorithms presented in this paper aim to reduce the search space automatically. They will automatically create a relevant subset of features utilizing a wrapped classifier fitness function. A significant increase in the classification ability of a Bayesian classifier in crater detection using image texture features.

2 Related Work

There have been many methods used to automatically detect craters. R. Honda [4] built a SOM using Hough transforms to extract geometric features. They then perform best parameter selection to reduce duplicate detections using a genetic algorithm based on the center location and radius of the detected crater. In this work we use genetic algorithms to choose the best features to build a classifier with which is different from R. Honda's method.

This paper uses standard classification methods to determine the probability that an image is in fact a crater as inspired by W. Ding [3] and L. Banderia [1]. Y. Cheng [2]. They used the concept of a confidence evaluation to detect craters and J. Kim [5] used a fitness check to determine if the candidate was a crater or not.

This work utilizes Haar features to perform candidate image classification. Haar features were used by W. Ding [3] L. Banderia [1], and S. Liu [7] and are the state of the art in crater detection because of their adaptive and discriminative ability. They are used in this work as crater features.

3 Genetically Enhanced Feature Selection

This section presents three genetic algorithms used for feature subset selection. Each algorithm builds upon the one before it in an attempt to achieve better results. The first algorithm is explained in detail and then only modifications are explained for the next two algorithms. First the main concepts of genetic algorithms, genetic representation, and fitness are discussed. For each algorithm there is an initial plan, explanation of steps, and a complexity analysis.

The three proposed methods of genetically enhanced feature selection are shown in Algorithms 3, 4, and 5. These vary in the way feature subsets are chosen to be crossed over. The goal is to pick the best feature subsets so that when they are combined will generate a feature subset with a higher fitness score than either of the original. The first algorithm attempts to choose the best two feature subsets and use them as parents while the later algorithms attempt to introduce controlled randomness. Controlled randomness is introduced by randomly selecting from feature subsets that are weighted based on their fitness score. Later simulated annealing is used to introduce more randomness at the beginning of the algorithm.

3.1 Genetic Representation

In these algorithms the genetic representation is a subset of features that are used in building a classifier. This is referred to in this work as a feature subset but is also called an individual or chromosome. The representation is treated as a subset and as a vector. This is achieved using the concept of a bitvector to set features as on or off. Each index of the bitvector represents one feature. The contents of a subset are the on features represented by the bitvector. The magnitude of the subset is the sum of all possible features.

3.2 Wrapped Classifier Fitness Function

The fitness function used in the following algorithms is modeled as an evaluation function in the F1 search space. It can be said that the fitness function is wrapped around a classifier. The F1 fitness metric, $fitness = F1 = \frac{2}{\frac{1}{recall}+\frac{1}{Precision}}$, takes into account two important attributes of a classifier, precision and recall. This allows us to compare two classifiers using one value. Using the F1 measure also allows the priority queue datastructure to be used. The calculation for precision, $Precision = \frac{truepositives}{truepositives+falsepositives}$, takes into account how generously the classifier predicted something was a positive example. This metric fails to describe the situation when the classifier has ignored many positive examples. Recall, $Recall = \frac{truepositives}{truepositives+falsenegatives}$, fixes this problem by describing the ability of the classifier to predict all the positive examples correctly. It can be thought of as describing the coverage of classifier. Recall by itself cannot describe the classifier because it does not take into account the case where the classifier marked everything as a positive instance.

3.3 Random Crossover

Instead of splitting the feature subset somewhere in the middle so the order is preserved at a loss for feature equality [9], we use random crossover to ensure that element of the feature subset has an equal chance of being preserved. This change is due to the absence of order that is involved with features.

The random crossover used, as shown in Algorithm 1, is the process of merging two parent subsets to make a new child subset. This resulting child subset is composed of the parent subsets. This process is used to simulate the mixing of chromosomes during natures genetic process. If a feature is enabled in both parents then it will be enabled in the child. If it varies in the parents then there is a 50% chance it will be preserved in the child. If a feature is turned off in both parents it will be turned off in the child. If this method was used exclusively in the algorithms then the children would converge to a local or global maximum fitness. This convergence is not desired because we want to avoid local maximums so the feature subsets are also mutated.

3.4 Mutation

Mutation is used to avoid the convergence of algorithms at a local maximum. As shown in Algorithm 2; mutation involves randomly flipping a percentage of bits in the feature subset to enable or disable features. Mutation as described here takes a percentage as an argument and mutates that percentage of the feature subset. This is used to simulate natures genetic mutations.

Algorithm 1: Preform Random Crossover $f1 \bigotimes f2$

Input: Feature Subset Vector $f1$
Feature Subset Vector $f2$
Output: Feature Subset Vector $f3$

for $|f1|$ **do**
  if $0.5 < Random(0, 1)$ **then**
    $f3_i = f1_i$
  else
    $f3_i = f2_i$

Algorithm 2: Preform Random Mutation $M(f1, \delta)$

Input: Feature subset $f1$
Percentage to mutate δ
Output: Feature subset vector $f2$

for δ *of* $|f3|$ **do**
  $r = Random(0, |f3|)$ // random index
  $f3_r = \neg f3_r$

3.5 Highest Fitness (Greedy) Selection

Also called GHF (Genetic Highest Fitness), the initial plan for this algorithm was to greedily select the best two feature subsets. This would then concentrate the randomness to the crossover and mutation phases. The expectation is that combining good feature subsets will produce better feature subsets.

Algorithm 3: Highest Fitness (Greedy) Selection (GHF)

Input: Features Γ
Iterations I
Inital random subsets s
Percentage to mutate m
Maximum size of Υ *size*
Output: Feature subset Γ'

Add a full instance of Γ to Υ
Add s random subsets of Γ to Υ
for *each* i *in* I **do**
  $\{f1 | f \in \Upsilon, fitness(f1) \geq fitness(f)\}$
  $\{f2 | f \in (\Upsilon \cap \neg\{f1\}), fitness(f2) \geq fitness(f)\}$
  $f3 = f1 \bigotimes f2$ // crossover subset
  $f3' = M(f3, m)$ // mutate subset m percent
  $\{\Upsilon | f \in \Upsilon \cap \{f3'\}\}$
$\{\Gamma' | \Gamma' \in \Upsilon, f \in \Upsilon, fitness(\Gamma') \geq fitness(f)\}$

The GHF steps are explained now. Step 1: seed the algorithm with a subset that contains all features. Step 2: seed the algorithm with an initial set of feature subsets. Step 3: we loop some number of times to simulate many generations of evolution. Step

4: we select a parent from the set of feature subsets that has the best fitness score. Step 5: remove the already selected feature subset and then select the feature subset with the highest fitness score. Step 6: randomly crossover f1 and f2 to create f3. Step 7: mutate m percent of this new feature subset. Step 8: define upsilon to contain the mutated f3. Step 9: define Gamma prime to be a feature subset that has the highest fitness score in upsilon.

3.6 Weighted Random Selection

Algorithm 4: Weighted Random Selection

Input: Features Γ
Iterations I
Inital random subsets s
Percentage to mutate m
Maximum size of Υ $size$
Output: Feature subset Γ'

Add a full instance of Γ to Υ
Add s random subsets of Γ to Υ
for *each* i *in* I **do**
    $f1 = GetWeightedSubset(\Upsilon)$
    $f2 = GetWeightedSubset(\Upsilon \cap \neg\{f1\})$
    $f3 = f1 \bigotimes f2$ // crossover subset
    $f3' = M(f3, m)$ // mutate subset m percent
    $\{\Upsilon | f \in \Upsilon \cap \{f3'\}\}$
$\{\Gamma' | \Gamma' \in \Upsilon, f \in \Upsilon, fitness(\Gamma') \geq fitness(f)\}$

$GetWeightedSubset$:
Input: Current set of feature subsets Υ
Output: Feature subset $f2$
$fn \in \Upsilon$
$\{\hat{fn} | \sum_n fn = 1\}$
$r = Random(0, 1)$
foreach $\hat{fn}$ **do**
    $r = r - \hat{fn}$
    if $r < 0$ **then**
        $returnf$

Also called GWR (Genetic Weighted Random), the initial plan for this algorithm was to increase the chance that a feature subset with better complementing features would be chosen. This is implemented using a weighting method. The feature subsets are selected at random but weighted based on their fitness values. To explain this sample data is shown in Figure 2. The sample data points are the values 0.95, 0.90, 0.80 and 0.70. To the right of the image those values have been scaled to 0.284, 0.269, 0.239, and 0.209. This allows a random number between 0 and 1 to select a feature. Figure 1 shows the result of the highest fitness score removed. The fitness values are normalized again to handle this change so that a new random number between 0 and 1 will select a feature subset.

Algorithm 4 has it's feature subset selection method modified. Steps 3 and 4 are changed now to call a GetWeightedSubset method. In GetWeightedSubset Step 10: defines a sub routine. Step 11: defines that f_n is a element of upsilon. Step 12: produces a normalized version of the feature subset and a fitness value to be used below only. Step 13: generates an r between 0 and 1 that we will subtract from to pick a feature subset. Step 14: loops for each feature subset. Step 15: subtracts the current feature subsets normalized fitness value from r. Step 16+17: if r has gone below 0 then we select that feature subset.

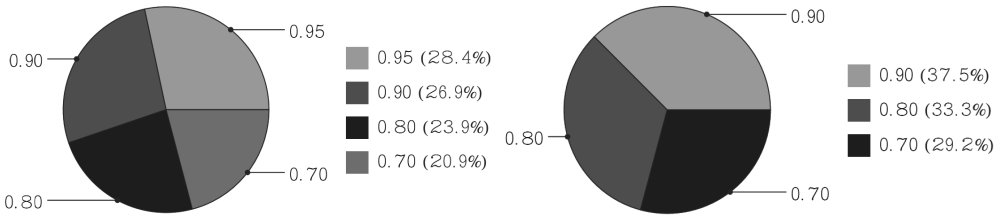

Fig. 1. Left: Sample values weighted based on fitness score, Right: Sample values weighted based on fitness score after one feature subset is removed

3.7 Weighted Random Selection with Simulated Annealing

Also called GWRSA (Genetic Weighted Random w/ Simulated Annealing), the idea for this algorithm is to initially weight all feature subsets the same during feature subset selection and then gradually weight them by their fitness score as the number of iterations increases.

As shown in Algorithm 5 the GetWeightedSubset method has been modified to take the current iteration as an argument. Steps 1 and 2 use this iteration value to drive the weighting to be close to equal at the beginning and properly weighted when iterations are at the end.

Algorithm 5: GetWeightedSubset w/ Simulated Annealing

Input: Current set of feature subsets Υ
Input: Current iteration i
Output: Feature subset $f2$

1 $annealing = I - i$
2 $\{\hat{fn} | \sum_n fn + annealing = 1\}$
3 $r = Random(0, 1)$
4 **foreach** $\hat{fn}$ **do**
5 $r = r - \hat{fn}$
6 **if** $r < 0$ **then**
7 $returnf$

3.8 Complexity

GHF is the most efficient algorithm out of the three. GWR and GWRSA have an added penalty due to the way they use randomness to avoid local maximums. The complexity of GHF is $O(ic\Gamma)$ where i is the number of iterations, c is the complexity of the classifier used to calculate the fitness score, and Γ is the number of features used. In GWR the change from selecting the feature subset with the highest fitness score to weighting and selecting causes the algorithm to increase in complexity. All the fitness values must now be added to create a normalization term. This increases the complexity to $O(ic\Gamma^2)$. A limit on the number of feature subsets in Γ would reduce the complexity but that method is not used in this algorithm. The complexity of GWRSA does not increase the complexity of GWR because the only change is an addition during the computation of the normalization term.

4 Experimental Results

This section analyses the advantages of using these feature selection algorithms. This is done by creating a super set of features that contain discriminate and indiscriminate features. The goal of the feature selection algorithm will be to remove the indiscriminate features and return a feature subset with only discriminate ones.

A challenge of experimenting with these algorithms is finding their optimal parameters. This section will also analyze the mutation rate, number of iterations, maximum number of feature subsets accumulated, and the number of feature subsets generated for the initial pool. This section will also compare this algorithm to a variety of other algorithms applied to the same dataset.

In these experiments the **training set** consists of 166 positive examples of craters and 343 negative examples of ground without cratering are used from the HRSC h0905_0000 nadir panchromatic image. This training set used is selected to simulate crater detection and was inspired by the ones used by W. Ding [3] and L. Banderia [1]. Positive examples contain craters centered and cropped as shown in Figure 2. This training set provides the ability to analyze the algorithms without dealing with the size and complexity of applied crater detection.

To evaluate the proposed algorithms Haar **features** are used in combination with a Bayesian classifier. Haar features have a proven ability to detect craters [3]. A Bayesian classifier is used because of it's naive use of all features. Haar features were first proposed by Papageorgiou [8], then applied to face detection by Viola and Jones [13], and then applied to crater detection by W. Ding [3]. Haar features are described using feature masks that specify white and black regions. The masks are overlaid on the crater image and the sum of each region's pixel values are calculated and then the difference is taken. In Figure 3 the masks above A are basic Haar feature masks. The masks above B and C are horizontally and vertically scaled to capture contrast and texture that will not fit into a square. The features extracted depend on the image format for precision and size. Haar features can be optimized for speed using a technique discussed by Viola and Jones [13] that allows for O(1) calculations of Haar features from an image that has had a corresponding Integral Image computed.

In the following experiments 58 Haar features are used. Figure 3 shows the outlines of these features to specify the coverage area. They were chosen to provide a challenge to the classifier while still providing discriminating features.

The **initial pool size** is the set of feature subsets that are given to the algorithm to start the process. There needs to be two or more feature subsets to start. Values from

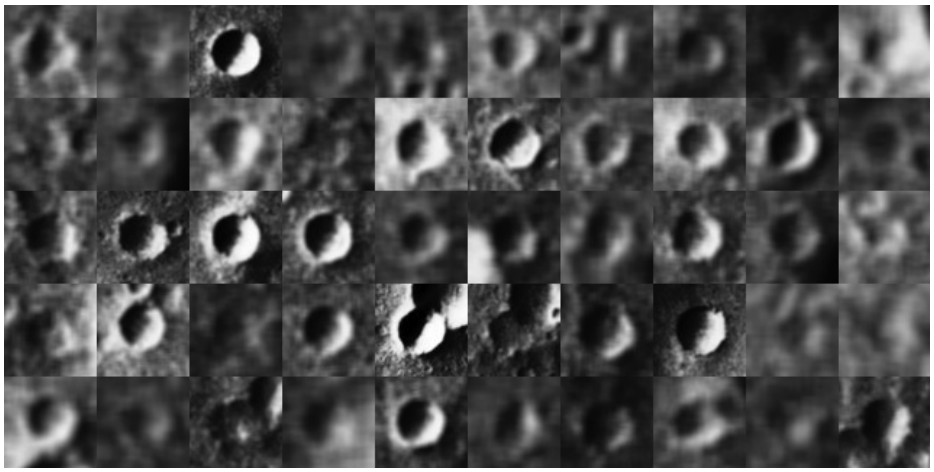

Fig. 2. Images used for positive examples

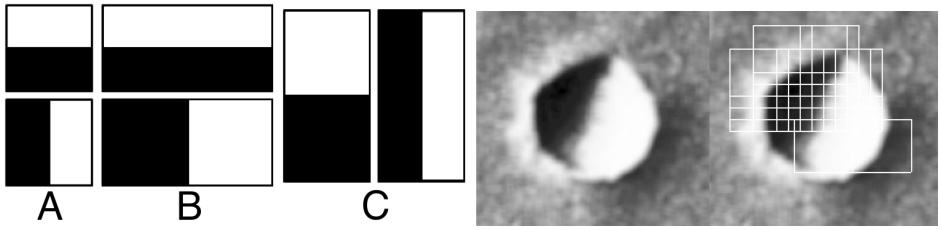

Fig. 3. Left: Haar Feature Masks Used, Right: Coverage of Haar Features on Crater Image

10 to 700 are used over 1000 iterations at 5% mutation to determine the optimal value. There does not seem to be any advantage to varying this parameter.

The **maximum feature subsets accumulated** variable is the limit of feature subsets that will be maintained in memory during the program execution. This is only used for the Weighted Random selection and the Weighted Random Simulated annealing feature selections. Values are sampled from 3 to 1000 during 10000 iterations at 5% mutation. The scale starts at 3 because otherwise it is the Highest Fitness Score feature selection. The Highest Fitness Score feature selection method keeps only 2 feature subsets in memory so there is no collection of feature subsets to vary. Figure 4 shows that the optimal values appears to be around 10.

A **mutation** rate needs to be chosen for GHF, GWR, and GWRSA. The rate is the percentage of the feature subset that will be randomly turned on or off. A constant percentage is used for every iteration. The elements that are changed are randomly selected each iteration. An experiment was performed using 10,000 iterations, 10 randomly generated initial feature subsets, and a Naive Bayes Classifier. In Figure 5, 5% is shown to be the best mutation rate.

The **number of iterations** used would be a limiting factor in the application of these algorithms. Figure 5 shows all three proposed algorithm's fitness score grouped by iterations but varying in configurations.

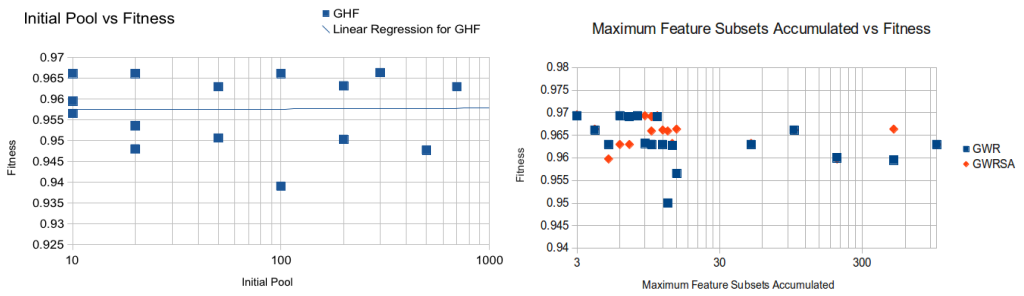

Fig. 4. Left: Initial Pool Sized vs Fitness, Right: Maximum Feature Subsets Accumulated vs Fitness

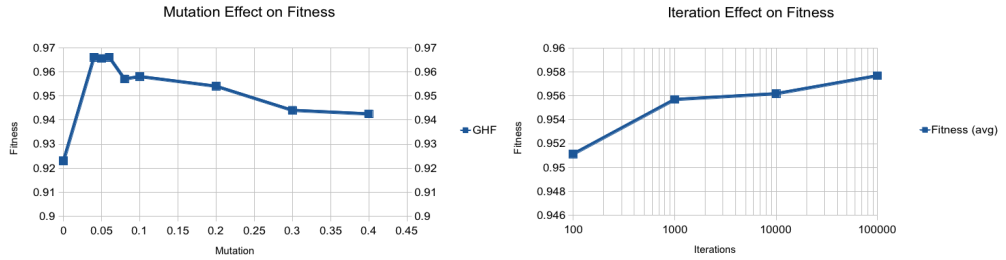

Fig. 5. Left: Average Percentage of Mutation Effectiveness, Right: Average Iteration Effect on Fitness

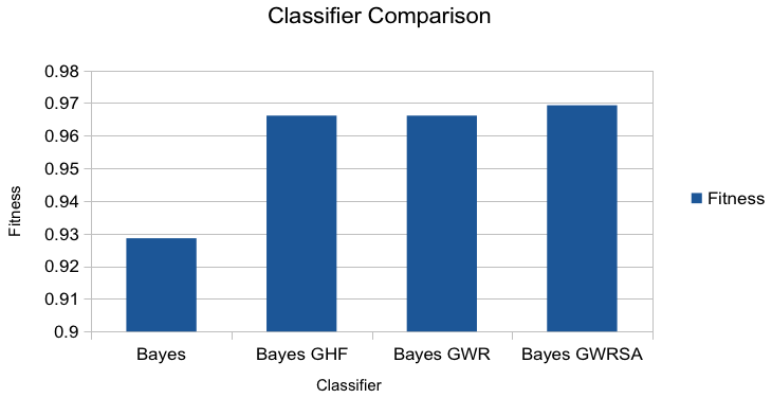

Fig. 6. Comparison of Classifiers

A **classifier comparison** is shown in Figure 6. The best performance of these algorithms is compared to the standard Naive Bayes classifier result. The algorithms always start with the standard Naive Bayes result because they use all the features which ensures the result will never decline. The results show that genetically enhanced feature selection offers a significant increase in the classification ability to the standard Naive Bayes classifier.

5 Conclusion

This paper presented three feature selection algorithms that increase the classification ability of the Naive Bayes classifier. This is necessary because during applications of machine learning the classifier is presented with discriminate and indiscriminate features. This increase in classification ability is caused by training the classifier with a subset of features containing discriminate features. This algorithm is shown to boost the classification ability of a classifier that does not perform feature selection itself.

Acknowledgment

This research was supported in part by NASA Grant NNX09AK86G and NSF Grant 1062749.

References

1. L. Bandeira, W. Ding, and T. F. Stepinski. Automatic detection of sub-km craters using shape and texture information. In *Proceedings of the 41st Lunar and Planetary Science Conference*, The Woodlands, Texas, Mar. 2010.
2. Y. Cheng, A. E. Johnson, L. H. Matthies, and C. F. Olson. Optical landmark detection for spacecraft navigation. *Advances in the Astronautical Sciences*, 114:1785–1803, 2003.
3. W. Ding, T. F. Stepinski, Y. Mu, L. Bandeira, R. Ricardo, Y. Wu, Z. Lu, T. Cao, and X. Wu. Sub-kilometer crater discovery with boosting and transfer learning. *ACM Transactions on Intelligent Systems and Technology*, 2(4), July 2011.
4. R. Honda, Y. Iijima, and O. Konishi. Mining of topographic feature from heterogeneous imagery and its application to lunar craters. *Progress in Discovery Science*, pages 27–44, 2002.
5. J. R. Kim, J. Muller, S. van Gasselt, J. G. Morley, G. Neukum, et al. Automated crater detection, a new tool for mars cartography and chronology. *Photogrammetric engineering and remote sensing*, 71(10):1205, 2005.
6. B. Leroy, G. Medioni, E. Johnson, and L. Matthies. Crater detection for autonomous landing on asteroids. *Image and Vision Computing*, 19(11):787–792, 2001.
7. S. Liu, W. Ding, J. P. Cohen, D. Simovici, and T. F. Stepinski. Bernoulli trials based feature selection for crater detection. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, Illinois, Nov. 2011. ACM.
8. C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. *Center for Biological and Computational Learning Artificial Intelligence Laboratory*, 1998.
9. S. Russell. *Artificial intelligence*. Pearson Education, Upper Saddle River N.J. ;;Harlow, 3rd ed. edition, 2009.
10. G. Salamuniccar and S. Loncaric. Open framework for objective evaluation of crater detection algorithms with first test-field subsystem based on MOLA data. *Advances in Space Research*, 42(1):6–19, July 2008.
11. E. R. Urbach and T. F. Stepinski. Automatic detection of sub-km craters in high resolution planetary images. *Planetary and Space Science*, 57(7):880–887, 2009.
12. T. Vinogradova, M. Burl, and E. Mjolsness. Training of a crater detection algorithm for mars crater imagery. In *2002 IEEE Aerospace Conference*, Mar. 2002.
13. P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.