Chapter 6
Feature Selection and Heterogeneous Descriptors

Various image features were described and their invariances and repeatabilities were explored in Chap. 4. The question we would like to answer in this chapter is, if d features are to be selected from among $D > d$ features for inclusion in an image descriptor, which features should be chosen? Creation of image descriptors from homogeneous features was discussed in the previous chapter. Our focus in this chapter will be to create descriptors from heterogeneous features.

Feature selection is the problem of reducing the number of features in a recognition or matching task. Feature selection problems arise in regression analysis, independence analysis, discriminant analysis, cluster analysis and classification [3], inductive learning [15, 21], and image matching, which is of particular interest in image registration.

In regression analysis, features that add little to the regression accuracy are discarded. Independent analysis is the problem of determining whether a D-dimensional structure can be represented exactly or approximately by $d < D$ dimensions, and if so, which of the original dimensions should be used. In discriminant analysis, the objective is to find whether some features can be dropped without significantly changing the discrimination power of a recognition system. In clustering and classification also, there is a need to remove features that do not influence the final clustering or classification result.

In inductive learning, since a learner uses all available features, the presence of irrelevant information can decrease the learning performance [39]. Therefore, again, there is a need to remove the irrelevant features to improve the learning performance. For example, if feature x_2 is a linear function of feature x_1 or if $x_2 = x_1 + \varepsilon$, where ε represents random noise, then either x_1 or x_2 can be dropped with little change in the classification result.

Given a set of D features from an image or a window, the feature selection problem is that of selecting a subset $d < D$ of features that contains more discriminatory information than any other subset of d features. An exhaustive search for the optimal solution requires $\binom{D}{d}$ comparisons, which is prohibitively large even for moderate values of D and d.

If the features are statistically independent, the problem may be solved by selecting individually the d best features out of D [26]. This involves first selecting the feature that maximizes a matching/recognition criterion. Among the remaining features, the next feature is then selected that maximizes the same criterion, and the process is repeated until the best d features are selected. However, Elashoff et al. [12] and Cover [6] show that the set of best two features do not necessarily contain the two best features. Toussaint [41] shows that the set of best two features may not even include the best feature. Therefore, the set of most informative d features out of D is not necessarily the set of the d individually most informative features. We would like to select the features in such a way as to achieve the smallest classification/matching error.

It is generally expected that the selected features (1) be independent of each other, and (2) when combined, provide the most complete information about the objects to be recognized. Condition 1 ensures that the selected features do not contain redundant information and condition 2 ensures that the selected features will least ambiguously distinguish different objects (or in our case, image windows) from each other.

Fourier transform and Zernike moments satisfy these conditions if a sufficient number of them are selected, but we know these features are not very efficient when used in recognition/matching. One can often find a combination of features that are not independent (orthogonal) but can provide more complete information about the objects or image windows. It has also been shown that informative class-specific features provide a higher discriminative power than generic features such as Zernike moments or wavelets [43]. We, therefore, will relax the orthogonality requirement and select rich features that may have some overlap with other features but when combined can provide the most complete information about the objects to be recognized or the windows to be located in an image.

Given M windows $\{O_i : i = 1, \dots, M\}$ and D features $\{f_i : i = 1, \dots, D\}$ from each window, we would like to determine the smallest subset d among D features that can distinguish the windows from each other with least ambiguity. We start by extracting features that are the least sensitive to changes in scale, orientation, contrast, noise, and blurring from the windows. Among the extracted features, we then select a subset that can best distinguish the windows from each other.

If the feature vector with d components is used to describe the contents of a window, the feature vector can be represented by a point in a d-dimensional space. By increasing the number of features, distance between the points in the feature space will increase, improving the recognition/matching accuracy. However, as the number of dimensions increases, the computational resources needed to solve a recognition/matching problem increase also. The key is to find the smallest d that can solve the problem with a required error tolerance.

Since it is possible for a pattern to appear repeatedly in an image, and knowing that a descriptor provides only information about a pattern local to a window, some kind of an information exterior to the window is needed to distinguish the pattern in different locations from each other. Assuming windows are taken centered at the control points in an image, examples of such exterior information are:

(1) distance of a control point to the control point closest to it, (2) the degree of a control point in the minimum-spanning tree (MST) or triangulation of the points, and (3) the largest/smallest angle between edges of the MST or triangulation at the control point. One or more of these exterior features, which are invariant to image scale and orientation, can be used together with the interior features of a window to distinguish windows containing the same pattern in different locations in an image from each other.

In the rest of this chapter, various feature selection methods are reviewed and their uses in creation of efficient heterogeneous image descriptors are explored.

6.1 Feature Selection Algorithms

Feature selection algorithms generally follow either a filter algorithm or a wrapper algorithm [8, 9, 21], although hybrid algorithms that take advantage of the strengths of both have been proposed also [27]. A filter algorithm selects features without regard to the final matching rate. For example, it may remove features that are highly dependent and retain only features that are independent. In a wrapper algorithm, features are selected taking into consideration the final matching outcome and will rely on the particular correspondence algorithm used. Wrapper algorithms are generally more accurate than filter algorithms, but they are computationally costlier.

Hybrid algorithms combine the strengths of the filter and wrapper algorithms and work well when a large feature set is provided. In a hybrid algorithm, first the best feature subset of a given cardinality is selected by removing redundant, irrelevant, or dependent features. Then by a wrapper algorithm and cross-validation, the final best features are selected [8, 18, 32, 47, 52].

6.1.1 Filter Algorithms

Given feature set $X = \{x_i : i = 1, \ldots, D\}$, we would like to select the feature subset $Y = \{y_j : j = 1, \ldots, d\}$ from X that is least redundant. Das [7] (also see [42]) measured redundancy by calculating linear dependency, while Heydorn [14] measured redundancy by calculating statistical dependency. A feature z in X is considered redundant with respect to feature subset Y if the probability distributions of (Y, z) and (Y) completely overlap. That is,

$$P\big[F(Y, z) = F(Y)\big] = 1, \tag{6.1}$$

where $F(Y)$ denotes the probability distribution of features in Y and $F(Y, z)$ denotes the cumulative distribution of Y and z. By adding feature z to feature subset Y, the distribution of Y does not change, and so Y still carries the same information. Koller and Sahami [22] remove redundant features in such a way that the class-conditional probability after removal of the redundant features is as close as possible to the class-conditional probability of the original features.

In an attempt to remove redundant features in a feature set, King [19] and Jolliffe [16] used a cluster merging idea. Initially, each feature is included in a cluster of its own. Then, the two closest clusters are merged to a new cluster. The cluster feature is then considered the average of the two features. The merging process is repeated by letting the feature representing the merged cluster be the average of the features in it. The process is stopped any time the desired number of clusters or features is reached. This method, in effect, merges similar features, producing features that are less similar. Note that this method transforms given features to new features while reducing redundancy.

Based on the clustering idea of King [19] and Jolliffe [16], Mitra et al. [30] used feature similarity to subdivide a feature set into clusters in such a way that the features in a cluster are highly similar, while those in different clusters are highly dissimilar. Similarity is measured using Pearson correlation. If N windows are available in the reference image, each with an associating feature vector, the correlation between features x any y is calculated from

$$r(x, y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x \sigma_y} \tag{6.2}$$

where

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i, \tag{6.3}$$

and

$$\sigma_x = \left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2\right)^{1/2}, \qquad \sigma_y = \left(\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2\right)^{1/2}. \tag{6.4}$$

Clustering of the features is achieved by the k-nearest neighbor (k-NN) method. First, the k-nearest features of each feature are identified and among them the feature with the most compact subset, determined by its distance to the farthest neighbor, is selected. Then, those k-neighboring features are discarded. Initially, k is set to $D - d$ and is gradually reduced until distance of a feature to the kth feature closest to it becomes larger than a required tolerance ε or k reaches 1.

Note that when Pearson correlation is used to measure the similarity between two features, the measure only detects linear dependency between features. If the features have nonlinear dependencies, a measure such as information gain or mutual information should be used.

Information gain is the amount of reduction in entropy of a feature x after observing feature y. That is,

$$IG(x|y) = H(x) - H(x|y), \tag{6.5}$$

where $H(x)$ is the entropy of feature x and is computed from

$$H(x) = -\sum_{i=1}^{N} P\big(x(i)\big)\log_2\big(P\big(x(i)\big)\big). \tag{6.6}$$

$P(x(i))$ denotes the probability that feature x will have value $x(i)$ and $H(x|y)$ is the entropy of observing feature x after having observed feature y and is computed from

$$H(x|y) = -\sum_{j=1}^{N} P\big(y(j)\big) \sum_{i=1}^{N} \big[P\big(x(i)|y(j)\big) \log_2\big(P\big(x(i)|y(j)\big)\big)\big]. \tag{6.7}$$

Given N windows in the reference image with associating feature vectors, the information gain for any pair of features can be calculated in this manner.

If two features are highly dependent, the obtained information gain will be smaller than when two less dependent features are used. Therefore, information gain can be used as a means to remove dependent (redundant) features from a set. Yu and Liu [48–50] normalized information gain $IG(x|y)$ with respect to the sum of the entropies of x and y to obtain a *symmetrical uncertainty* measure:

$$SU(x, y) = 2\left[\frac{IG(x|y)}{H(x) + H(y)}\right] = 2\left[\frac{IG(y|x)}{H(x) + H(y)}\right]. \tag{6.8}$$

This measure varies between 0 and 1, with 0 indicating that x and y are completely independent and 1 indicating that x and y are completely dependent. The closer the symmetrical uncertainly between two features is to 1, the more dependent the features will be, so one of them should be removed from the feature set. This process can be repeated until all features are sufficiently independent of each other. Note that accurate calculation of entropies for $H(x)$ and $H(y)$ as well as the conditional entropy $H(x|y)$ requires well populated probability distributions for $P(x)$, $P(y)$, and $P(x|y)$, and that requires a large number of reference windows.

Dependency can be measured using generalized Shannon mutual information [11] also:

$$I(x, y) = H(x) + H(y) - H(x, y), \tag{6.9}$$

where $H(x)$ and $H(y)$ are entropies of features x and y and $H(x, y)$ is the joint entropy of features x and y, defined by

$$H(x, y) = -\sum_{i=1}^{N}\sum_{j=1}^{N} P\big(x(i), y(j)\big) \log_2 P\big(x(i), y(j)\big). \tag{6.10}$$

$P(x(i), y(j))$ is the probability that feature x has value $x(i)$ and feature y has value $y(j)$. Bonnlander and Weigend [5], Wang et al. [44], and Bell and Wang [4] used mutual information to measure feature dependency and relevance. Ding, Peng, and Long [10, 33, 34] considered relevance the inverse of redundancy and selected features in such a way that redundancy was minimized.

To select a representative subset from a set, Wei and Billings [45] first selected the feature that correlated with most features in the set. They then added features one at a time such that each newly selected feature correlated the most with the remaining features and correlated the least with features already selected. Least correlation was ensured through an orthogonalization process. They argue that if many features correlate with feature x, feature x represents those features and so should be included in the selected subset.

6.1.2 Wrapper Algorithms

Wrapper algorithms select features in a training step where correspondence between windows in the images are known. Given a set of corresponding windows in two images, a wrapper algorithm selects the features such that the number of incorrect matches is minimized. The matching process can be considered a nearest-neighbor classifier. To determine the window in the reference image that corresponds to a window in the sensed image, distances between all windows in the reference image to the window in the sensed image are determined and the reference window closest to the sensed window is selected as the corresponding window. Examples of distance measures are given in (5.12) and (5.13).

Perhaps the simplest wrapper algorithm is the *Max-Min* algorithm [2], which selects the features one at a time until the required number of features is reached.

Max-Min Algorithm Given features $X = \{x_i : i = 1, \dots, D\}$ from each window and knowing the correspondence between windows in reference and sensed images, we would like to find feature subset $Y \subset X$, which is initially empty, and upon exit contains d features maximizing the number of correspondences. Let $J(x_i)$ show the number of correspondences obtained when using feature x_i, and let $J(x_i, x_j)$ show the number of correspondences obtained when using features x_i and x_j, where $i \neq j$. Then:

1. Select feature x_j in X and include in Y where $J(x_j) = \max_i(J(x_i))$.
2. Among the remaining features in X, select feature x_k and include in Y if for all features x_j in Y we obtain $J(x_k, x_j) = \max_k\{\min_j(\Delta J(x_k, x_j))\}$, where $\Delta J(x_k, x_j) = J(x_k, x_j) - J(x_j)$.
3. Repeat Step 2 until Y contains d features.

$\Delta J(x_k, x_j)$ is the increase in the number of correspondences by moving feature x_k from X to Y. The feature in X that maximizes the minimum increase in the number of correspondences when considered pairwise against all features in Y is selected and added to Y. The computational complexity of this algorithm is on the order of Dd operations, where each operation involves a few additions, multiplications, and comparisons.

A feature selection algorithm that is based on the search algorithm of Marill and Green [28] removes the least relevant features from X one at a time until the desired number of features remains. This algorithm is known as sequential backward selection (SBS) [20]. Similarly, an algorithm developed by Whitney [46] selects the most relevant features one at a time from a feature set to create the desired subset. This algorithm is known as sequential forward selection (SFS) [20].

SBS starts from the given feature set and removes the worst feature from the set at each iteration until the desired number of features remains. On the other hand, SFS starts from an empty subset, selects a feature from among the given set and moves it to the subset in such a way that the number of correspondences is maximized. Steps in the SFS and SBS algorithms are given below [40].

Sequential Forward Selection (SFS) Algorithm Given N corresponding windows in two images and feature set $X = \{x_i : i = 1, \ldots, D\}$ calculated for each window, we would like to select feature subset $Y = \{y_j : j = 1, \ldots, d\}$, such that $d < D$, $Y \subset X$, and the obtained feature subset maximizes the number of correct correspondences.

1. Choose a dissimilarity measure to determine the distance between two feature vectors.
2. Compute the number of correspondences using each feature individually. Move the feature from X to Y that produces the most correct correspondences, and let $k = 1$.
3. Select a feature from among the remaining $D - k$ features. There are $D - k$ such cases. Use the selected feature and the k features already in Y to determine the number of correct correspondences. Move the feature that produces the most correspondences from X to Y and incremented k by 1.
4. If $k = d$, return Y. Otherwise, go to Step 3.

Sequential Backward Selection (SBS) Algorithm Starting from feature subset Y containing all D features in X, remove features from Y one at a time until d features remain and the d features produce the smallest number of incorrect correspondences.

1. Choose a dissimilarity measure to determine the distance between two feature vectors.
2. Let Y contain all D features in X and $k = D$.
3. Eliminate one feature from Y. There are k possibilities. Find the number of correspondences obtained when using the remaining $k - 1$ features. Eliminate the feature that produces the minimum number of incorrect correspondences when using the remaining features in Y, and decrement k by 1.
4. If $k = d$, return the d features in Y. Otherwise, go to Step 3.

SFS starts from an empty subset and adds one feature at a time to it until the required number of features is reached, and SBS starts from the entire set and removes features one at a time from it until the required number of features remain.

The choice of forward or backward comes from the relative value of d with respect to D. If d is close to D, backward elimination is more efficient than forward selection. However, if d is very small compared to D, forward selection is more efficient than backward elimination.

Note that in backward elimination, once a feature is removed from the set of features, it never gets a chance to return to the set. Also, in forward selection, once a feature is included in the subset, it remains there and there is no chance for it to get out. This problem, which is known as *nesting*, may introduce considerable redundancy in the created subset.

To reduce redundancy in backward elimination, Pudil et al. [35] described a *floating algorithm* that revises SBS algorithm to exchange a feature already excluded from the subset with a feature in the subset if that increases the number of correspondences. This involves only a slight modification of the SBS algorithm. Insert a step between Steps 3 and 4 in the SBS algorithm as follows: Replace each of the k features remaining in the set, with one of the features already removed from the set. There are $D - k$ such replacements for each of the k features. If a replacement increases the number of correspondences, keep the replacement. The SFS can be revised in the same manner to avoid nesting. Insert a step between Steps 3 and 4 in SFS algorithm as follows: Replace each of the k features in Y with one of the $D - k$ features still remaining in X. If this replacement increases the number of correspondences, keep it.

Generalized versions of SFS and SBS have also been proposed [20]. In the generalized SFS (GSFS), if k features are already selected, all subsets with $k + r$ features are generated by adding r features from the remaining $D - k$ features to the existing subset. Then the subset that maximizes the number of correspondences is selected. In the generalized SBS (GSBS), if k features remain in the set after removing features from the set, all combinations of $k + r$ subsets are created by adding to the subset all combinations of r features out of $D - k$. Again, the subset maximizing the number of correspondences is chosen. As r, the number of features added or removed in each iteration, is increased, a more optimal subset is obtained but at a higher computational cost. Aha and Bankert [1] explored other variations of the SFS and SBS algorithms.

Somol et al. [37] further improved the feature selection optimality by combining the forward and backward steps, calling the new algorithm *adaptive floating feature selection*. By carrying out a more thorough search in each selection step, a solution closer to the optimal one is reached, at a higher computational cost.

An SFS algorithm that prevents nesting is proposed by Michael and Lin [29]. The idea is generalized by Stearns [38] into an algorithm appropriately called *plus l take away r*, which adds l features to the subset at each iteration while removing r features.

Plus l Take Away r Algorithm Assuming $l > r$, X represents a set of D features, and $Y \subset X$ represents the feature subset containing $d < D$ features:

1. Choose a distance measure and initially let Y be an empty set. Also let $k = 0$.
2. Choose feature x in X and add to Y if new Y maximizes the number of correspondences, then increment k by 1. If $k = d$, return Y. Otherwise, repeat this step l times.
3. Remove that feature y from Y such that the new Y minimizes the number of incorrect correspondences, then decrement k by 1. Repeat this step r times.
4. Repeat Steps 2 and 3 in sequence as many times as needed until d features are obtained in Y.

Stearns [38] compared the computational complexities of plus-l-take-away-r and SFS and SBS algorithms, finding that the computational complexity of plus-l-take-away-r algorithm is only several times higher than those of the SFS and SBS algorithms. Kittler [20] found that the plus-l-take-away-r algorithm produces significantly better results than the SFS and SBS algorithms but only when $l \approx r$. When l and r are too different, the process is incapable of avoiding nesting.

The above algorithms are suboptimal. To obtain the optimal solution, exhaustive search is needed, but that is not feasible when the feature set contains 100 or more features. A branch-and-bound algorithm developed by Narender and Fukunaga [31] selects the optimal feature subset without explicitly evaluating all possible feature subsets. The algorithm, however, requires the feature selection criterion be monotonic. That is, the provided features should be such that the correspondence algorithm would produce more correspondences when using $d+1$ features than when using d features.

A branch-and-bound algorithm repeatedly partitions the solution space into smaller subspaces. Within each subspace, a lower bound is found for the solutions there. Those subspaces with bounds that exceed the cost of a feasible solution are excluded from future partitioning. Among all remaining subspaces, the subspace with the lowest cost is partitioned and the process is repeated until a solution is reached. The solution reached first will be the lowest-cost as any solution obtained subsequently will have a higher cost [25].

Narender and Fukunaga [31] used the branch-and-bound principle to develop an optimal feature selection algorithm that selects the feature subset maximizing the number of correspondences. The performance of the branch-and-bound algorithm is further improved by Yu and Yuan [51] by expanding the tree branches that are more likely to be a part of the solution subset. The algorithm adds one feature at a time to a subset, always keeping track of the subset producing the highest number of correspondences. The process will find the highest number of correspondences for a required number of features. The steps of this algorithm are as follows.

Branch-and-Bound Algorithm

1. Create the root of a tree T and an empty list L. Also, let $m=0$ and $k=0$. m shows the height of a node from the root and k shows the index of a selected feature. 0 implying that no feature has been selected.
2. Create $i=1,\dots,D-d+1$ nodes, then save in the ith node x_i and the number of correspondences found using feature x_i. Also, increment m by 1 and save that at the nodes. Next, make a link from each newly created node to the root and save in L pointers to the created nodes in the descending order of the number of obtained correspondences. These nodes represent the leaves of the tree at height $m=1$.
3. Take the first node in L and suppose the index of the node is k. If $m=d$, backtrack from that node to the root and return the indices of the features saved at the visited nodes. Otherwise continue.
4. Create $l=D-d-k+m+1$ nodes pointing to node k in the tree and save at the created nodes feature indices $k+1,\dots,k+l$. Also, find the number of correspondences found for each node using the features from that node to the root and save that number at the node. Then, increment m by 1 and save m at the created nodes. Next, update L by inserting the newly created nodes in the list in such a way that the list remains ordered in descending order of the number of correspondences. L now points to nodes in the tree that represent the leaves of the new tree. Go back to Step 3.

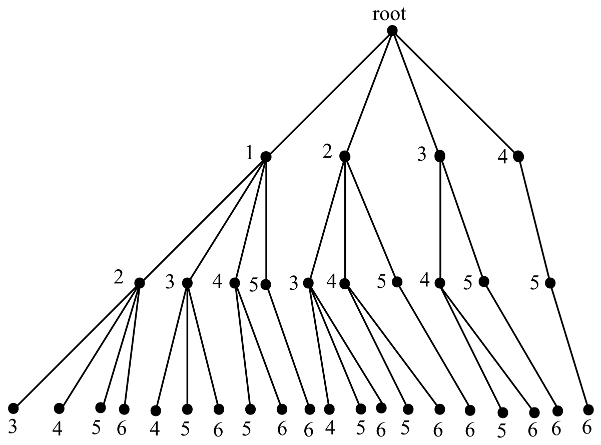

Fig. 6.1 The complete solution tree for selecting 3 features out of 6 by the branch-and-bound algorithm. The solution subset is found by partially traversing the tree from *top* to *bottom* while maximizing the number of correspondences

In Step 2, instead of creating D nodes only $D - d + 1$ nodes are created. Because d features are required, the distance of a node at level 1 to a leaf of the tree should be d to produce d features. After using d features, there will be $D - d$ features remaining, each creating a new branch in the tree and each branch defining the solution subset. This results in $D - d + 1$ nodes at level 1. At each level in the tree, the leftmost node has $D - d + 1$ branches, but the branch to the right of it at the same level has one fewer branches. This is because the order of features in a set is not important. For example, once set $\{x_1, x_2\}$ is tested, there is no need to test $\{x_2, x_1\}$.

Step 3 returns an optimal subset containing d features, because if a path exists that can produce a higher number of correspondences, it is already found. We know this because the path with the highest number of correspondences is always on top of list L, and we know that our matching algorithm produces more correspondences with feature subset $(Y \cup \{x_l\})$ than with feature subset Y, where x_l is any feature that is not already in Y. The tree structure when X contains 6 features and Y is required to contain 3 features is shown in Fig. 6.1. Note that use of list L makes it possible to find the solution set without exploring the entire tree.

The above algorithm shows steps of a forward algorithm as the number of features is gradually increased until the desired number of features is obtained. A similar algorithm can be designed to eliminate features one at a time while minimizing the number of incorrect correspondences until the required number of features is obtained. It has been shown that the branch and bound algorithm works quite well even when the feature selection criterion is not monotonic [13].

A study carried out by Kittler [20] in a two-class problem with 20 features finds that the optimal branch-and-bound algorithm has the highest discriminatory power, closely followed by the GSFS and GSBS algorithms, and the discriminatory power of the two get closer as the number of features added to or removed from the subset increases in each iteration. The Max-Min algorithm is found to be the fastest but the least accurate when compared to the branch-and-bound and the GSFS and GSBS algorithms.

Although the ultimate goal of a feature selector is to take the fewest features to achieve a required recognition rate, a property of a feature selector that is important is its stability [17, 23, 24, 36]. If features are extracted of the same phenomenon under slightly different conditions, will the feature selector under consideration select the same features? The stability can be measured using the Hamming distance between selected features in two tests. Stability shows robustness and ensures similar results under small perturbations in data.

6.2 Selecting the Best Features for Image Matching

Tables 4.1 and 4.2 in Chap. 4 identify the most invariant and repeatable features. If the features are to be selected individually, the tables are useful. However, if a combination of d features is to be selected for image matching, we do not know which d features to choose.

A set of features performing well on a class of images may perform poorly on another class of images. To determine optimal features in image matching, a training step is required that finds the optimal features for a class of images. Given two representative images and a number of corresponding windows in the images, we would like to find the set of d features that can find most corresponding windows in the images?

We start with the features suggested in Tables 4.1 and 4.2. That is, we assume the features available to us are $X = \{x_1$: mean intensity (L_2), x_2: second-order moment 1 (L_{23}), x_3: second-order moment 2 (L_{24}), x_4: third-order moment 2 (L_{26}), x_5: normalized complex moment invariant of order $(1, 1)$ (L_{50c}), x_6: Beaudet's cornerness measure (L_{64}), x_7: local frequency domain entropy (L_{75}), x_8: steerable filter response (L_{78}), x_9: correlation response to Laws mask B_{11} (L_{86a}), x_{10}: correlation response to Laws mask B_{22} (L_{86b}), x_{11}: smoothed intensity with a Gaussian of standard deviation 2 pixels (L_{88}), x_{12}: Laplacian after smoothing with a Gaussian of standard deviation 2 pixels (L_{90}), x_{13}: center contrast (L_{96}), and x_{14}: fractal dimension (L_{111})$\}$.

If different windows in the same image contain the same pattern, we will distinguish them from each other by using two features that provide external information about the windows. As external information, we will use x_{15}: MST degree and x_{16}: triangulation degree. The degree of a control point is considered the number of edges connected to the control point in the MST or in the triangulation of the control points. Local features are measured within circular windows of radius r pixels centered at the points. In the following experiments, r is taken to be 8 pixels.

Given a set of corresponding control points in two images, we take a circular window of radius $r = 8$ pixels centered at each control point and determine the above-mentioned 16 features using intensities in the window. As control points, we choose local extrema of the response of the Laplacian of Gaussian (LoG) of standard deviation 2 pixels in an image. Then we calculate the 16 features for windows centered at each control point. In the following, we will then determine the smallest subset $d < 16$ features that can determine the most correspondences. Sequential

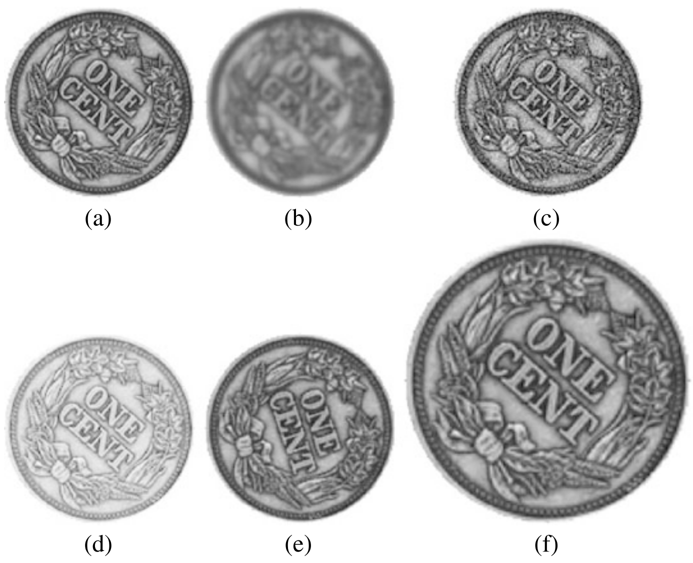

Fig. 6.2 (**a**) Image of a coin, used as the reference image. The reference image after (**b**) blurring, (**c**) addition of noise, (**d**) histogram equalization, (**e**) rotation, and (**f**) scaling

forward selection (SFS) algorithm will be used to search for the best suboptimal features.

To demonstrate the search process, the simple coin image shown in Fig. 6.2a is used as the reference image. The image is then blurred with a Gaussian of standard deviation 1.5 pixels to obtain the image shown in Fig. 6.2b. Next, Gaussian noise of standard deviation 20 is added to the reference image to obtain the noisy image shown in Fig. 6.2c. Values higher than 255 are set to 255 and values below 0 are set to 0. The reference image after histogram equalization is shown in Fig. 6.2d. This can be considered a nonlinear but monotone intensity transformation of the reference image. Finally, we rotate the reference image by 30° clockwise to obtain the image in Fig. 6.2e, and scale the reference image by a factor of 1.5 to obtain the image in Fig. 6.2f.

The 100 strongest and well dispersed points determined in these images by the LoG detector are shown in Fig. 6.3. Details about point detection are provided in Chap. 3. The problem to be solved is to locate the points of the reference image in images (b)–(f) using the 16 or fewer features calculated within circular windows of radius 8 pixels centered at the points.

Knowing the geometric relations between image (a) and images (b)–(f) in Fig. 6.2, the coordinates of corresponding points in the images will be known. Therefore, given 100 points in each image, we know which point in image (a) corresponds to which point in any of the other images. Mapping the points in image (a) to the space of images (b)–(f), we obtain images (b)–(f) shown in Fig. 6.4. The points in image (a) are shown in red, while those in images (b)–(f) in Fig. 6.3 are shown in green. points that are perfectly aligned in these images appear in yellow.

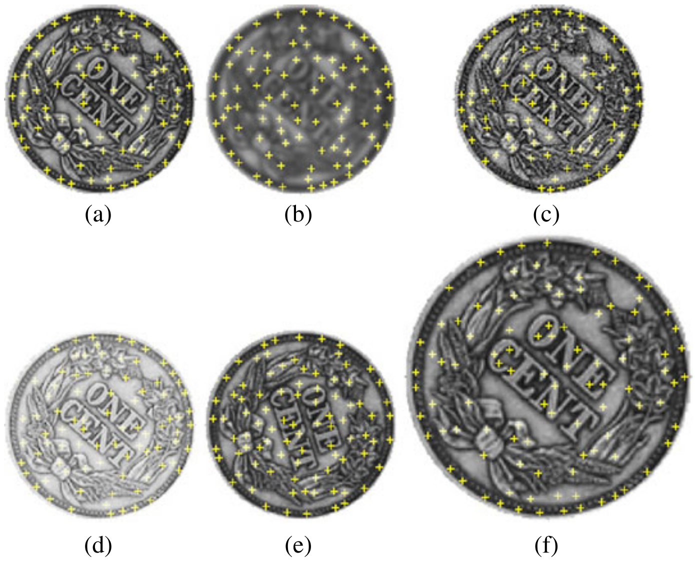

Fig. 6.3 (**a**)–(**f**) Points detected in images (a)–(f) in Fig. 6.2 using the LoG detector of standard deviation 2 pixels. Only the 100 strongest and well dispersed points are used in the experiments

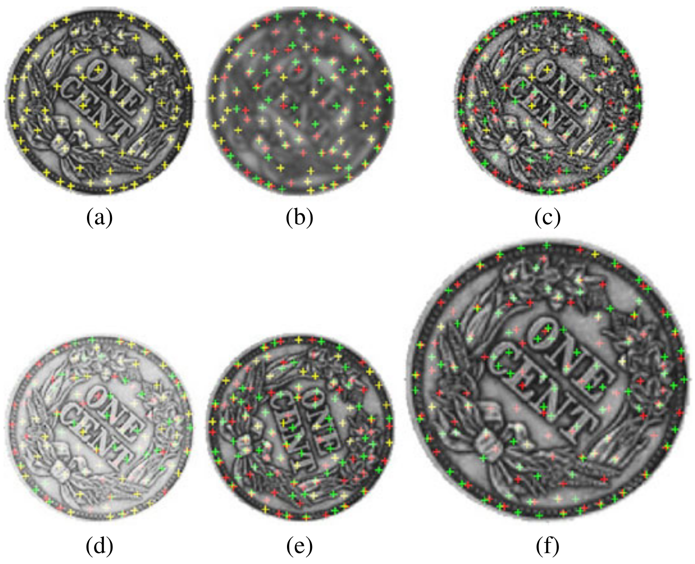

Fig. 6.4 (**a**) Same as image (a) in Fig. 6.3. (**b**)–(**f**) Points in image (**a**) when overlaid with images (b)–(f) in Fig. 6.3. In these images, points of image (**a**) are shown in *red*, while points in images (b)–(f) in Fig. 6.3 are shown in *green*. Perfectly aligned points appear as *yellow*

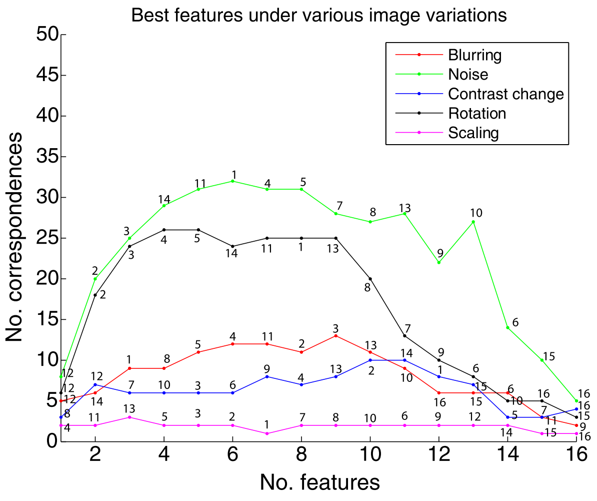

Fig. 6.5 Performances of various image features in image matching under various image changes. The numbers along a plot indicate the order in which the particular features are selected as the number of features is increased

Out of 100 points in these images, 66, 60, 68, 48, and 28 of the points in blurred image (b), noisy image (c), contrast enhanced image (d), rotated image (e), and scaled image (f) in Fig. 6.3 are the same as those in the image (a). We consider points that are within distance $\sqrt{2}$ of each other as corresponding points to compensate for the digital errors caused by image rotation and scaling.

Knowing the true correspondence between points in these images, we will take circular windows of radius 8 pixels centered at each point, calculate the 16 features listed above from each window, and through feature selection determine the best feature subsets that can find the most correspondences between image (a) and images (b)–(f) in Fig. 6.2.

The matching results are summarized in Fig. 6.5. The graph shows the number of correspondences obtained for a particular set of features obtained by the SFS algorithm. As the number of features increases, the number of correspondences initially increases up to a point, but then it decreases as more features are added. The features selected early appear to contain the right information for a particular image variation and improve matching result. However, features selected later in the process contain misleading information and instead of improving the matching, worsen the process, increasing the number of mismatches.

Under blurring, the best features are (1) x_{12}: Laplacian of Gaussian intensity, (2) x_{14}: fractal dimension, (3) x_1: mean intensity, (4) x_8: steerable filter response, (5) x_5: normalized complex moment 2, (6) x_4: third-order moment 2, (7) x_{11}: smoothed Gaussian intensity, (8) x_2: second order moment 1, and (9) x_3: second-order moment 2. These features not only remain relatively unchanged under image blurring or change in resolution, they contain sufficient discriminatory power to recognize various patterns.

The worst features under image blurring are x_9: correlation response to Laws mask B_{11} and x_7: local frequency domain entropy. These features appear to be very sensitive to image blurring and should be avoided when matching images obtained at different resolutions. 66 of the 100 landmarks detected in the coin image and its

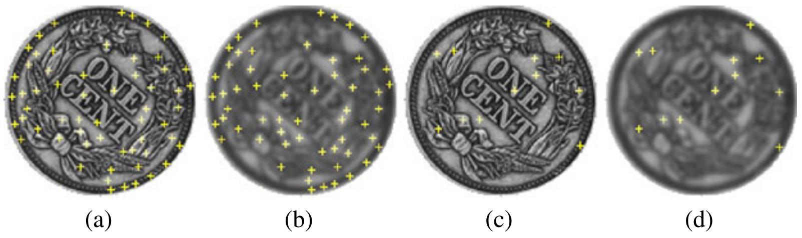
Fig. 6.6 (**a**), (**b**) 66 true corresponding points existing between the coin image and its blurred version. (**c**), (**d**) 13 of the correspondences are found when using the first 9 features as selected by the SFS algorithm

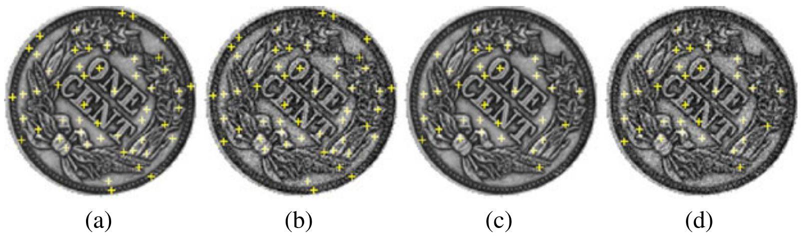
Fig. 6.7 (**a**), (**b**) 60 true corresponding points exist in the coin image and its noisy version. (**c**), (**d**) 32 of the 60 correspondences are found using the first 6 features as selected by the SFS algorithm

blurred version truly correspond to each other. This is depicted in Fig. 6.6a and b. Two points are considered corresponding points if they appear at the same pixel location or at adjacent pixel locations. 13 of the 66 correspondences are found when using the above 9 features as depicted in (c) and (d) in Fig. 6.6. Use of more features actually reduces the number of correspondences as can be observed in the graph in Fig. 6.5.

Under noise, the most effective feature are: (1) x_{12}: Laplacian of Gaussian intensity, (2) x_2: second-order moment 1, (3) x_3: second-order moment 2, (4) x_{14}: fractal dimension, (5) x_{11}: Gaussian smoothed intensity, and (6) x_1: mean intensity. These feature remain relatively unchanged under noise. They also carry the most information when matching the noisy images used in this experiment. The worst features under noise are x_{16}: triangulation degree and x_{15}: MST degree. Since noise can change the number of points and their locations, the triangulation and the MST of the points will differ, making these features unsuitable for matching noisy images. 60 out of the 100 points in the coin image an in its noisy version, as shown in the images in Figs. 6.7a, b are the same. Using the first 6 features as selected by the SFS algorithm, 32 of the correspondences are found as shown in Figs. 6.7c, d. Use of more features reduces the number of correspondences.

Under monotone intensity transformation, the most effective features for image matching are found to be: (1) x_8: steerable filter response, (2) x_{12}: Laplacian of Gaussian intensity, (3) x_7: local frequency domain entropy, (4) x_{10}: correlation re-

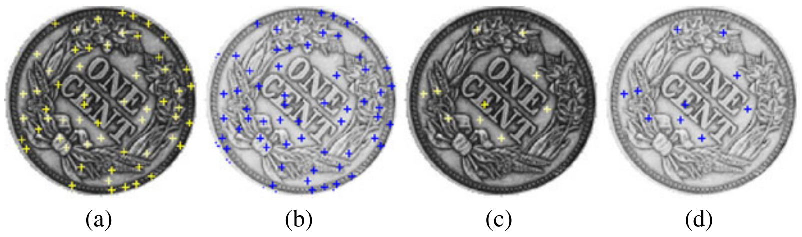

Fig. 6.8 (**a**), (**b**) 68 corresponding landmarks are detected in the coin image and its contrast enhanced version. (**c**), (**d**) 10 of the correspondences are identified using the first 10 features as selected by the SFS algorithm

sponse to Laws mask B_{22}, (5) x_3: second order moment 2, (6) x_6: Beaudet's cornerness measure, (7) x_9: correlation response to Laws B_{11} mask, (8) x_4: third-order moment 2, (9) x_{13}: center contrast, and (10) x_2: second-order moment 1. Observing the graph in Fig. 6.5, we see that the number of correspondences rises to 7 and then falls to 6 and again rises to 10. This behavior is a side effect of the suboptimal feature selection algorithm of SFS.

The worst features under monotone change in image contrast are x_{16}: triangulation degree, x_{11}: Gaussian smoothed intensity, and x_5: normalized complex moment. 68 features out of the 100 points in the coin image and its contrast enhanced version truly correspond to each other as shown in Figs. 6.8a, b. Using the first 10 features as selected by the SFS algorithm the process finds 10 of the correspondences as shown in Figs. 6.8c, d.

Images that have rotational differences are best matched using features (1) x_{12}: Laplacian of Gaussian intensity, (2) x_2: second-order moment 1, (3) x_3: second-order moment 2, and (4) x_4: third-order moment 2. These features are known to be rotation invariant. Among all rotationally invariant features, these features capture sufficient non-overlapping information to enable effective recognition/matching of local neighborhoods. The worst features to be used in matching of images with rotational differences are x_{15}: MST degree and x_{16}: triangulation degree.

It is interesting to note that the MST and triangulation degrees are not very reliable under image rotation. Although MST and triangulation of a set of points are invariant to rotation of the points, the point detector that has been used has found only 48 corresponding points out of 100 in the coin image and its rotated version as shown in Figs. 6.9a, b. As a result, the MST and the triangulation of control points in the images look very different from each other, producing very little useful information for matching. The process has been able to find 26 of the correspondences using the first 4 features as detected by the SFS algorithm.

Finally, examining the matching result of images with scaling differences, we find that the best features are: (1) x_4: third-order moment 2, (2) x_{11}: Gaussian smoothed intensity, and (3) x_{13}: center contrast. The worst features are: x_{16}: triangulation degree, x_{15}: MST degree, and x_1: mean intensity. It is interesting to see that under change in scale, mean intensity is not a reliable feature for use in image matching. Not knowing the scale difference between two images, averaging a fixed

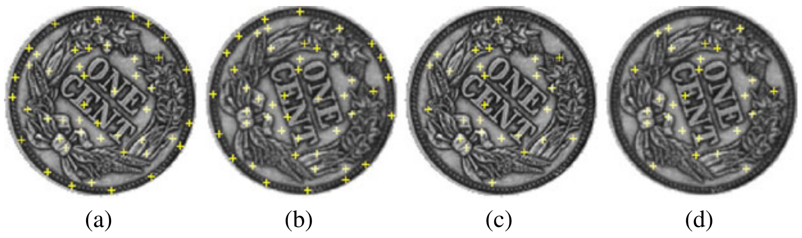

Fig. 6.9 (**a**), (**b**) 48 true corresponding landmarks exist between the coin image and its rotated version. (**c**), (**d**) 26 of the correspondences are found when using the first 4 features as selected by the SFS algorithm

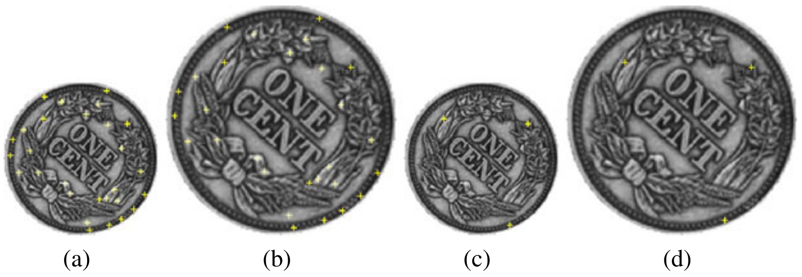

Fig. 6.10 (**a**), (**b**) 28 true corresponding points are present in the coin image and its scaled version. (**c**), (**d**) 3 of the correspondences are identified when using the first 3 features as selected by the SFS algorithm

neighborhood of 3 × 3 pixels centered at a point results in considerably different intensities that are not reliable in image matching.

The point detector has been able to find only 28 common points out of 100 in both images. Using the first 3 features as detected by the SFS algorithm, 3 of the correspondences are identified (Fig. 6.10). Using more features actually reduces the number of correspondences as the remaining features do not contain useful information when the images to be matched have unknown scaling differences.

Overall, worst matching results are obtained when the images have unknown scaling differences. A very small number of correspondences are obtained when considering all combinations of features. This can be attributed to (1) sensitivity of the features to change in scale, and (2) sensitivity of the point detector to change in scale. If a sufficiently large number of the same points is not detected in the images, the matching process cannot find a large number of correspondences.

Examining all these results, we see that various moments perform the best, and if the images do not have scaling difference, Laplacian of Gaussian intensity is also a very effective feature in image matching. It should be mentioned that the results obtained by the simple experiments above cannot be used to make a general conclusion about the power of the features. The performance of a feature can vary from image to image and it depends on the size of the window used in the calculations.

If the type of images to be registered is known, a number of the images may be used to identify those features that most effectively find the same points in the images and find correspondence between the points using the features of windows centered at the points. Once the best features are identified, they can be used in a custom system that can effectively register images in that class.

An important conclusion that can be reached from the above experiments is that use of more features does not necessarily mean a more accurate matching. Some features vary under certain image changes and do not contain useful information to help matching, and if used, they may confuse the matching process and reduce match rating.

When it comes to finding correspondence between windows in two images, in addition to the features of the windows, relations between the windows can be used as constraints to reject the false correspondences. Use of constraints in matching is discussed in detail in the next chapter.

References

1. Aha, D.W., Bankert, R.L.: A comparative evaluation of sequential feature selection algorithms. In: Proc. 5th Int'l Workshop Artificial Intelligence and Statistics, pp. 1–7 (1995)
2. Backer, E., Schipper, J.A.D.: On the max-min approach for feature ordering and selection. In: Seminar Series on Pattern Recognition, Liège University, Sart-Tilman, Belgium, pp. 2.4.1–2.4.7 (1977)
3. Beale, E.M.L., Kendall, M.G., Mann, D.W.: The discarding of variables in multivariate analysis. Biometrika **53**(3/4), 357–366 (1967)
4. Bell, D.A., Wang, H.: A formalism for relevance and its application in feature subset selection. Mach. Learn. **41**, 175–195 (2000)
5. Bonnlander, B.V., Weigend, A.S.: Selecting input variables using mutual information and non-parametric density estimation. In: Proc. International Symposium on Artificial Neural Networks (ISANN), pp. 42–50 (1996)
6. Cover, T.M.: The best two independent measurements are not the two best. IEEE Trans. Syst. Man Cybern. **4**(1), 116–117 (1974)
7. Das, S.K.: Feature selection with a linear dependence measure. IEEE Trans. Comput. **20**(9), 1106–1109 (1971)
8. Das, S.: Filters, wrappers and a boosting-based hybrid for feature selection. In: Proc. 18th Int'l Conf. Machine Learning, pp. 74–81 (2001)
9. Dash, M., Liu, H.: Feature selection for classification. Inell. Data Anal. **1**, 131–156 (1997)
10. Ding, C., Peng, H.C.: Minimum redundancy feature selection from microarray gene expression data. In: Proc. 2nd IEEE Conf. Computational Systems Bioinformatics, pp. 523–528 (2003)
11. Duncan, T.E.: On the calculation of mutual information. SIAM J. Appl. Math. **19**(1), 215–220 (1970)
12. Elashoff, J.D., Elashoff, R.M., Goldman, G.E.: On the choice of variables in classification problems with Dichotomous variables. Biometrika **54**, 668–670 (1967)
13. Hamamoto, Y., Uchimura, S., Matsunra, Y., Kanaoka, T., Tomita, S.: Evaluation of the branch and bound algorithm for feature selection. Pattern Recognit. Lett. **11**, 453–456 (1990)
14. Heydorn, R.P.: Redundancy in feature extraction. IEEE Trans. Comput. **20**(9), 1051–1054 (1971)
15. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Proc. 11th Int'l Conf. Machine Learning, pp. 121–129 (1994)

16. Jolliffe, I.T.: Discarding variables in a principal component analysis. I: Artificial data. J. R. Stat. Soc., Ser. C, Appl. Stat. **21**(2), 160–173 (1972)
17. Kalousis, A., Prados, J., Hilario, M.: Stability of feature selection algorithms. In: Proc. 5th IEEE Int'l Conf. Data Mining, pp. 218–225 (2005)
18. Ke, C.-H., Yang, C.-H., Chuang, L.-Y., Yang, C.-S.: A hybrid filter/wrapper approach of feature selection for gene expression data. In: IEEE Int'l Conf. Systems, Man and Cybernetics, pp. 2664–2670 (2008)
19. King, B.: Step-wise clustering procedures. J. Am. Stat. Assoc. **62**(317), 86–101 (1967)
20. Kittler, J.: Feature set search algorithms. In: Chen, C.H. (ed.) Pattern Recognition and Signal Processing, pp. 41–60 (1978)
21. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**, 273–324 (1997)
22. Koller, D., Sahami, M.: Toward optimal feature selection. In: Proc. 13th Int'l Conf. Machine Learning, pp. 284–292 (1996)
23. Křížek, P., Kittler, J., Hlaváč, V.: Improving stability of feature selection methods. In: Proc. 12th Int'l Conf. Computer Analysis of Images and Patterns, pp. 929–936 (2007)
24. Kuncheva, L.I.: A stability index for feature selection. In: Proc. 25th IASTED Int'l Multi-Conf. Artificial Intelligence and Applications, pp. 421–427 (2007)
25. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: A survey. Oper. Res. **14**(4), 699–719 (1966)
26. Lewis, P.M. II: Characteristic selection problem in recognition systems. IRE Trans. Inf. Theory **8**(2), 171–178 (1962)
27. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. Knowl. Data Eng. **17**(4), 491–502 (2005)
28. Marill, T., Green, D.M.: On the effectiveness of receptors in recognition systems. IEEE Trans. Inf. Theory **9**(1), 11–17 (1963)
29. Michael, M., Lin, W.-C.: Experimental study of information measure and inter-intra class distance ratios on feature selection and ordering. IEEE Trans. Syst. Man Cybern. **3**(2), 172–181 (1973)
30. Mitra, P., Murthy, C.A., Pal, S.K.: Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Anal. Mach. Intell. **24**(3), 301–312 (2002)
31. Narendra, P.M., Fukunaga, K.: A branch and bound algorithm for feature subset selection. IEEE Trans. Comput. **26**(9), 917–922 (1977)
32. Ng, A.Y.: On feature selection: Learning with exponentially many irrelevant features as training examples. In: Proc. 15th Int'l Conf. Machine Learning, pp. 404–412 (1998)
33. Peng, H., Ding, C., Long, F.: Minimum redundancy maximum relevance feature selection. In: IEEE Intelligent Systems, Nov./Dec., pp. 70–71 (2005)
34. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005)
35. Pudil, P., Novovičová, J., Kittler, J.: Floating search methods in feature selection. Pattern Recognit. Lett. **15**(11), 1119–1125 (1994)
36. Somol, P., Novovičová, J.: Evaluation stability and comparing output of feature selectors that optimize feature subset cardinality. IEEE Trans. Pattern Anal. Mach. Intell. **32**(11), 1921–1939 (2010)
37. Somol, P., Pudil, P., Novovičová, J., Paclík, P.: Adaptive floating search methods in feature selection. Pattern Recognit. Lett. **20**, 1157–1163 (1999)
38. Stearns, S.D.: On selecting features for pattern classifiers. In: 3rd Int'l Conf. Pattern Recognition, pp. 71–75 (1976)
39. Talavera, L.: Feature selection as a preprocessing step for hierarchical clustering. In: Proc. 16th Int'l Conf. Machine Learning, pp. 389–397 (1999)
40. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 4th edn. Academic Press, San Diego (2009), pp. 602, 605, 606
41. Toussaint, G.T.: Note on the optimal selection of independent binary features for pattern recognition. IEEE Trans. Inf. Theory **17**(5), 618 (1971)

42. Toussaint, G.T., Vilmansen, T.R.: Comments on feature selection with a linear dependence measure. IEEE Trans. Comput. **21**(4), 408 (1972)
43. Vidal-Naquet, M., Ullman, S.: Object recognition with informative features and linear classification. In: Proc. Int'l Conf. Computer Vision, pp. 281–288 (2003)
44. Wang, H., Bell, D., Murtagh, F.: Axiomatic approach to feature subset selection based on relevance. IEEE Trans. Pattern Anal. Mach. Intell. **21**(3), 271–277 (1999)
45. Wei, H.-L., Billings, S.A.: Feature subset selection and ranking for data dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 162–166 (2007)
46. Whitney, A.: A direct method for nonparametric measurement selection. IEEE Trans. Comput. **20**, 1100–1103 (1971)
47. Xing, E., Jordan, M., Karp, R.: Feature selection for high-dimensional genomic microarray data. In: Proc. 15th Int'l Conf. Machine Learning, pp. 601–608 (2001)
48. Yu, L., Liu, H.: Efficiently handling feature redundancy in high-dimensional data. In: Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 685–690 (2003)
49. Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: Proc. 20th Int'l Conf. Machine Learning, pp. 856–863 (2003)
50. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. J. Mach. Learn. Res. **5**, 1205–1224 (2004)
51. Yu, B., Yuan, B.: A more efficient branch and bound algorithm for feature selection. Pattern Recognit. **26**(6), 883–889 (1993)
52. Zhu, Z., Ong, Y.-S., Dash, M.: Wrapper-filter feature selection algorithm using a memetic framework. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **37**(1), 70–76 (2007)