As presented in the beginning of this section, this feature selection approach was developed for frontal human face recognition in natural upright position in image data. Interestingly, the first and second features selected in the process described by Algorithm 10.11 are easily interpretable. As Figure 10.27 demonstrates, the first feature corresponds to the frequent case when the eye region is darker than the region of the nose and cheeks. Similarly, the second feature is in agreement with the observation that the eyes are usually darker compared to the nose bridge located between them.

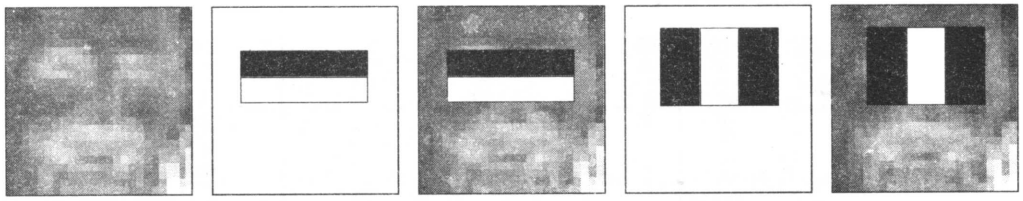


Figure 10.27: Two most significant feature determined by Algorithm 10.11 for face detection in subwindows of 24×24 pixels [Viola and Jones, 2001]. (a) Original image. (b) First most distinguishing feature corresponds to the frequent case when the eye region is darker than the region of the nose and cheeks. (c) The first feature overlaid over the original image. (d) Second most distinguishing feature is in agreement with the observation that the eyes are darker compared to the nose bridge located between them. (e) The second feature overlaid.

Once the previous step identifies the most distinguishing features, a cascade of classifiers is built to both decrease processing time and increase performance. The early-stage simple classifiers are set so that their false negative rates (number of missed detections) is close to zero. Of course, the price paid for such behavior is an increase in the false positive rate (number of detections not corresponding to true objects). However, the simpler early stage classifiers are used to quickly reject the majority of candidate locations (subwindows in which features are computed). The increasingly more complex classifiers are employed in the locations that remain unrejected. Ultimately, the remaining non-rejected locations are marked as the locations of identified objects.

Figure 10.28 shows this cascade of classifiers, a degenerate decision tree. For each location, classifier $n + 1$ is only called if the classifier n has not rejected the candidate location. The classifiers in all individual stages are trained using AdaBoost and adjusted to minimize false negatives. In the case of face detection, a powerful first stage classifier can be constructed from the two features identified above (Figure 10.27). This classifier detects 100% of face objects with about 40% false positives. The later-stage classifiers have increasingly higher false-positive rates–but since they are only applied to the subset of already-identified high-likelihood locations, the likelihood of invoking these classifiers is decreasing with the increased depth in the decision tree. The additional stages in the classifier cascade are added until the overall requirements on detection performance are met. As can be expected, since the features are evaluated at multiple scales and the subwindows are allowed to overlap, there may be multiple detections for each detected object, coming from the overlapping subwindows. Such multiple detections must be postprocessed to yield a single final detection per identified location in the image.

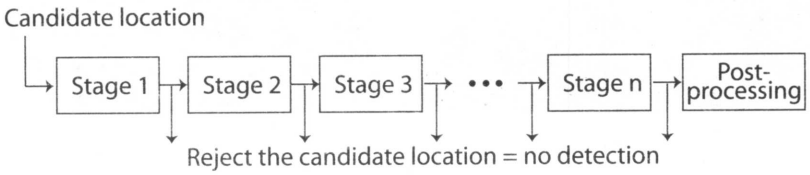
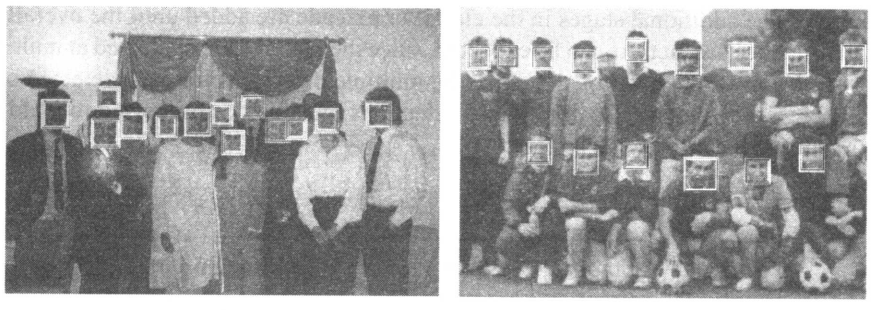


Figure 10.28: Detection cascade applied to each analyzed location–image subwindow. The early-stage simple classifiers reject less likely locations while maintaining a very low false negative rate. The later-stage more complex classifiers eliminate more and more false positives while they are set not to reject true positives.

In detecting faces, the complete system [Viola and Jones, 2001] consisted of 38 stages and used over 6,000 features. Despite these large numbers, the system only required about 10 feature evaluations per subwindow. As such, it was achieving high detection speed on average even when tested on a difficult data set with 75 million subwindows and over 500 faces present in the image data. Figure 10.29 shows examples of the training face images. Figure 10.30 shows face detection results obtained by the system. While a face detection problem was used to demonstrate the method, the approach itself is general and can be used for a variety of object detection and recognition tasks.

Figure 10.29: Some of the training face images used in the face detection system based on the boosted cascade of classifiers. *Courtesy of P. Viola, Microsoft Live Labs and M. Jones, Mitsubishi Electric Research Labs, ©2001 IEEE [Viola and Jones, 2001].*



(a) (b)

Figure 10.30: Example results of face detection using the described method of the boosted cascade classifiers. Each detected face is identified by an overlaying rectangle. *Courtesy of P. Viola, Microsoft Live Labs and M. Jones, Mitsubishi Electric Research Labs, ©2001 IEEE [Viola and Jones, 2001].*

10.7 SCENE LABELING AND CONSTRAINT PROPAGATION

Context plays a significant role in image understanding; the previous section was devoted to context present in pixel data configurations, and this section deals with semantic labeling of regions and objects. Assume that regions have been detected in an image that correspond to objects or other image entities, and let the objects and their interrelationships be described by a region adjacency graph and/or a semantic net (see Sections 4.2.3 and 9.1). Object properties are described by unary relations, and interrelationships between objects are described by binary (or n-ary) relations. The goal of scene labeling is to assign a label (a meaning) to each image object to achieve an appropriate image interpretation.

The resulting interpretation should correspond with available scene knowledge. The labeling should be consistent, and should favor more probable interpretations if there is more than one option. Consistency means that no two objects of the image appear in an illegal configuration—e.g., an object labeled *house* in the middle of an object labeled *lake* will be considered inconsistent in most scenes. Conversely, an object labeled *house* surrounded by an object labeled *lawn* in the middle of a *lake* may be fully acceptable.
Two main approaches may be chosen to achieve this goal.

- **Discrete** labeling allows only one label to be assigned to each object in the final labeling. Effort is directed to achieving a consistent labeling all over the image.
- **Probabilistic** labeling allows multiple labels to co-exist in objects. Labels are probabilistically weighted, with a label confidence being assigned to each object label.

The main difference is in interpretation robustness. Discrete labeling always finds either a consistent labeling or detects the impossibility of assigning consistent labels to the scene. Often, as a result of imperfect segmentation, discrete labeling fails to find a consistent interpretation even if only a small number of local inconsistencies is detected. Probabilistic labeling always gives an interpretation result together with a measure of confidence in the interpretation. Even if the result may be locally inconsistent, it often gives a better scene interpretation than a consistent and possibly very unlikely interpretation resulting from a discrete labeling. Note that discrete labeling may be considered a special case of probabilistic labeling with one label probability always being 1 and all the others being 0 for each object.
The scene labeling problem is specified by:

- A set of objects R_i, $i = 1,..., N$.
- A finite set of labels Ω_i for each object R_i (without loss of generality, the same set of labels will be considered for each object; $\Omega_i = \Omega j$ for any $i, j \in [1,..., N]$).
- A finite set of relations between objects.
- The existence of a compatibility function (reflecting constraints) between interacting objects.

To solve the labeling problem considering direct interaction of all objects in an image is computationally very expensive and approaches to solving labeling problems are usually based on **constraint propagation**. This means that local constraints result in local consistencies (local optima), and by applying an iterative scheme the local consistencies adjust to global consistencies (global optima) in the whole image.

Many types of relaxation exist, some of them being used in statistical physics, for example, simulated annealing (Section 9.6.2), and stochastic relaxation [Geman and Geman, 1984], etc. Others, such as **relaxation labeling**, are typical in image understanding. To provide a better understanding of the idea, the discrete relaxation approach is considered first.

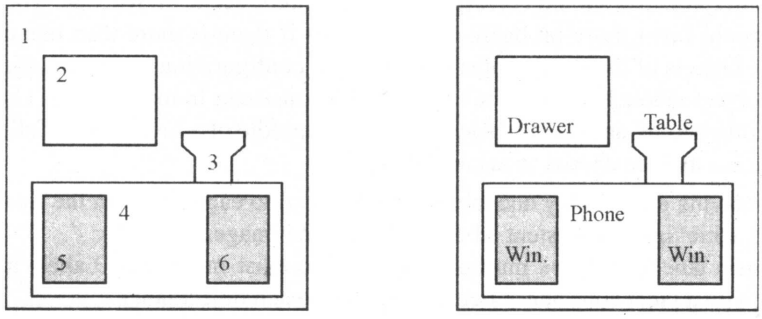
10.7.1 Discrete relaxation

Consider the scene shown in Figure 10.31a. Six objects are present in the scene, including the background. Let the labels be *background* (*B*), *window* (*W*), *table* (*T*), *drawer* (*D*), *phone* (*P*), and let the unary properties of object interpretations be (the example is meant to be illustrative only):

- A window is rectangular.
- A table is rectangular.
- A drawer is rectangular.

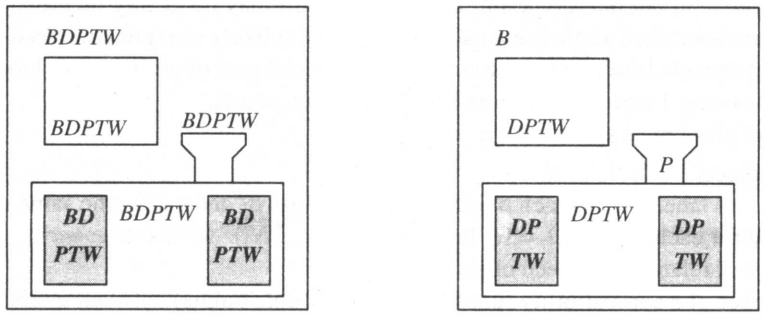
Let the binary constraints be:

- A window is located above a table.
- A phone is above a table.
- A drawer is inside a table.
- Background is adjacent to the image border.

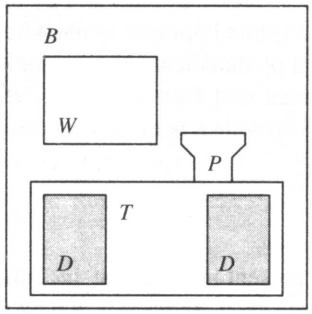


(a) (b)

Figure 10.31: Scene labeling. (a) Scene example. (b) Inconsistent labeling.



(a) (b)



(c)

B background
D drawer
P phone
T table
W window

Figure 10.32: Discrete relaxation. (a) All labels assigned to each object. (b) Locally inconsistent labels are removed. (c) Final consistent labeling.

Given these constraints, the labeling in Figure 10.31b is inconsistent. Discrete relaxation assigns all existing labels to each object and iteratively removes all the labels which may not be assigned to an object without violating the constraints. A possible relaxation sequence is shown in Figure 10.32.

At the beginning (Figure 10.32a), all labels are assigned to each object, and for each object all its labels are tested for consistency. Therefore, the label B can immediately be removed as inconsistent in objects 2, 3, 4, 5, and 6. Similarly, object 3 is not rectangular, therefore it violates the unary relation that must hold for *T*, *W*, D, etc.

The final consistent labeling is given in Figure 10.32c; note the mechanism of constraint propagation. The distant relations between objects may influence labeling in distant locations of the scene after several steps, making it possible to achieve a global labeling consistency of the scene interpretation although all the label-removing operations are local.

Algorithm 10.12: Discrete relaxation

1. Assign all possible labels to each object, considering the unary constraints.
2. Repeat steps 3–5 until global consistency is achieved or is found to be impossible.
3. Choose one object to update its labels.
4. Modify (delete inconsistent) labels of the chosen object by considering relations with other interacting objects.
5. If any object has no label, stop—a consistent labeling was not found.

The algorithm may be implemented in parallel with one difference: step 4 disappears as all objects are treated in parallel.

For a more detailed survey of discrete relaxation techniques, their properties, and technical difficulties that limit their applicability, see [Hancock and Kittler, 1990]. Although discrete relaxation is naturally parallel, a study of the complexity of discrete relaxation given in [Kasif, 1990] shows that a parallel solution is unlikely to improve known sequential solutions much. An interesting discrete relaxation control strategy using asynchronous activation of object updating actions (**daemons**) was introduced in [Barrow and Tenenbaum, 1976].

10.7.2 Probabilistic relaxation

Constraints are a typical tool in image understanding. The classical problem of discrete relaxation labeling was first introduced in [Waltz, 1957] in understanding perspective line drawings depicting 3D objects, and this problem is discussed briefly in Chapter 12. Discrete relaxation results in an unambiguous labeling; in a majority of real situations, however, it represents an oversimplified approach to image data understanding—it cannot cope with incomplete or imprecise segmentation. Using semantics and knowledge, image understanding is supposed to solve segmentation problems which cannot be solved by bottom-up interpretation approaches. Probabilistic relaxation may overcome the segmentation problems of missing objects or extra regions in the scene, but it results in an ambiguous image interpretation which is often inconsistent. It has been noted that a locally inconsistent but very probable (global) interpretation may be more valuable than a consistent but unlikely interpretation (e.g., a non-rectangular window located far above the table would be considered a phone in our example; this labeling would be consistent, even if very unlikely—see Figure 10.33).

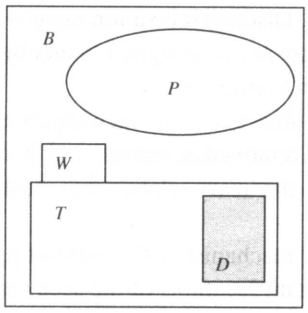


Figure 10.33: Consistent but unlikely labeling.

Probabilistic relaxation was introduced in [Rosenfeld et al., 1976] and has been used extensively in image understanding ever since. Consider the relaxation problem as specified above (regions R_i and sets of labels Ω_i) and, in addition, let each object R_i be described by a set of unary properties X_i. Similarly to discrete relaxation, object labeling depends on the object properties and on a measure of compatibility of the potential object labels with the labeling of other, directly interacting objects. All the image objects may be considered directly interacting, and a general form of the algorithm will be given assuming this. Nevertheless, only adjacent objects are usually considered to interact directly, to reduce computational demands of the relaxation. However, as before, more distant objects still interact with each other as a result of the constraint propagation. A region adjacency graph is usually used to store the adjacency information.

Consider the local configuration of objects given in Figure 10.34; let the objects R_j be labeled by θ_j; $\theta_j \in \Omega; \Omega = \{\omega_1, \omega_2, \ldots, \omega_R\}$. Confidence in the label θ_i of an object R_i depends on the configuration of labels of directly interacting objects. Let $r(\theta_i = \omega_k, \theta_j = \omega_l)$ represent the value of a compatibility function for two interacting objects R_i and R_j with labels θ_i and θ_j (the probability that two objects with labels θ_i and θ_j appear in a specific relation). The relaxation algorithm [Rosenfeld et al., 1976] is iterative and its goal is to achieve the locally best consistency in the entire image. The **support** q_j^s for a label θ_i of the object R_i resulting from the binary relation with the object R_j at the s^{th} step of the iteration process is

$$q_j^s(\theta_i = \omega_k) = \sum_{l=1}^{R} r(\theta_i = \omega_k, \theta_j = \omega_l)\, P^s(\theta_j = \omega_l), \tag{10.30}$$

where $P^s(\theta_j = \omega_l)$ is the probability that region R_j should be labeled ω_l. The support Q^s for the same label θ_i of the same object R_i resulting from all N directly interacting objects R_j and their labels θ_j at the s^{th} step is

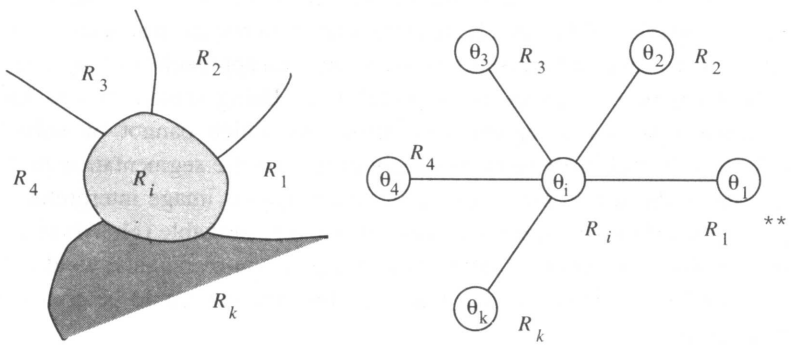


Figure 10.34: Local configuration of objects in an image—part of a region adjacency graph.

$$Q^s(\theta_i = \omega_k) = \sum_{j=1}^{N} c_{ij} q_j^s(\theta_i = \omega_k)$$
$$= \sum_{j=1}^{N} c_{ij} \sum_{l=1}^{R} r(\theta_i = \omega_k, \theta_j = \omega_l)\, P^s(\theta_j = \omega_l), \tag{10.31}$$

where c_{ij} are positive weights satisfying $\sum_{j=1}^{N} c_{ij} = 1$. The coefficients c_{ij} represent the strength of interaction between objects R_i and R_j. Originally [Rosenfeld et al., 1976], an updating formula was given which specified the new probability of a label θ_i according to the previous probability $P^s(\theta_i = \omega_k)$ and probabilities of labels of interacting objects,

$$P^{s+1}(\theta_i = \omega_k) = \frac{1}{K} P^s(\theta_i = \omega_k) Q^s(\theta_i = \omega_k), \tag{10.32}$$

where K is a normalizing constant

$$K = \sum_{l=1}^{R} P^s(\theta_i = \omega_l)\, Q^s(\theta_i = \omega_l). \tag{10.33}$$

This form of the algorithm is usually referred to as a **non-linear relaxation scheme**. A **linear scheme** [Rosenfeld et al., 1976] looks for probabilities such as

$$P(\theta_i = \omega_k) = Q(\theta_i = \omega_k) \quad \text{for all } i, k \tag{10.34}$$

with a non-contextual probability

$$P(\theta_i = \omega_k) = P(\theta_i = \omega_k | X_i) \tag{10.35}$$

being used only to start the relaxation process [Elfving and Eklundh, 1982].

A relaxation algorithm can also be treated as an optimization problem, the goal being maximization of the global confidence in the labeling [Hummel and Zucker, 1983]. The global objective function is

$$F = \sum_{k=1}^{R} \sum_{i=1}^{N} P(\theta_i = \omega_k) \sum_{j=1}^{N} c_{ij} \sum_{l=1}^{R} r(\theta_i = \omega_k, \theta_j = \omega_l)\, P(\theta_j = \omega_l) \tag{10.36}$$

subject to the constraint that the solution satisfies

$$\sum_{k=1}^{R} P(\theta_i = \omega_k) = 1 \quad \text{for any } i, \qquad P(\theta_i = \omega_k) \geq 0 \quad \text{for any } i, k. \tag{10.37}$$

Optimization approaches to relaxation can be generalized to allow n-ary relations among objects. A projected gradient ascent method [Hummel and Zucker, 1983] may be used to optimize equation (10.36), and an efficient version of this updating principle is introduced in [Parent and Zucker, 1989].

Convergence is an important property of iterative algorithms; as far as relaxation is concerned, convergence problems have not yet been satisfactorily solved. Although convergence of a discrete relaxation scheme can always be achieved by an appropriate design of the label updating scheme (e.g., to remove the inconsistent labels), convergence of more complex schemes where labels may be added, or of probabilistic relaxation, often cannot be guaranteed mathematically. Despite this fact, the relaxation approach may still be quite useful. Relaxation algorithms are one of the cornerstones of the high-level vision understanding processes, and applications can also be found outside the area of computer vision.

Relaxation algorithms are naturally parallel, since the label updating may be done on all objects at the same time. Many parallel implementations exist, and parallel relaxation does not differ in essence from the serial version. A general version is the following algorithm.

Algorithm 10.13: Probabilistic relaxation

1. Define conditional probabilities of interpretations (labels) for all objects R_i in the image [e.g., using equation (10.35)].
2. Repeat steps 3 and 4 until the best scene interpretation (a maximum of the objective function F) is reached.
3. Compute the objective function F [equation (10.36)], which represents the quality of the scene labeling.
4. Update probabilities of object interpretations (labels) to increase the value of the objective function F.

Parallel implementations of relaxation algorithms can be found in [Kamada et al., 1988; Dew et al., 1989; Zen et al., 1990].

Relaxation algorithms are still being developed. One existing problem with their behavior is that the labeling improves rapidly during early iterations, followed by a degradation which may be very severe. The reason is that the search for the global optimum over the image may cause highly non-optimal local labeling. A possible treatment that allows spatial consistency to be developed while avoiding labeling degradation is based on decreasing the neighborhood influence with the iteration count [Lee et al., 1989]. For a survey and an extensive list of references, see [Kittler and Illingworth, 1985; Kittler and Foglein, 1986; Kittler and Hancock, 1989]. A compact theoretical basis for probabilistic relaxation and close relations to the contextual classification schemes is given in [Kittler, 1987]. Improvements of algorithms for probabilistic relaxation can be found in [Lu and Chung, 1994; Christmas et al., 1996; Pelillo and Fanelli, 1997]. Application of the relaxation scheme to image segmentation is described in the next section.

10.7.3 Searching interpretation trees

Note that relaxation is not the only way to solve discrete labeling problems, and classical methods of **interpretation tree** searching may be applied. A tree has as many levels as there are objects present in the scene; nodes are assigned all possible labels, and a depth-first search based on back-tracking is applied. Starting with a label assigned to the first object node (tree root), a consistent label is assigned to the second object node, to the third object node, etc. If a consistent label cannot be assigned, a back-tracking mechanism changes the label of the closest node at the higher level. All the label changes are done in a systematic way.

An interpretation tree search tests all possible labelings, and therefore computational inefficiency is common, especially if an appropriate tree pruning algorithm is not available. An efficient method for searching the interpretation trees was introduced in [Grimson and Lozano-Perez, 1987]. The search is heuristically guided towards a *good* interpretation based on a *quality of match* that is based on constraints and may thus reflect feasibility of the interpretation. Clearly, an infeasible interpretation makes all interpretations represented down the tree infeasible also. To represent the possibility of discarding the evaluated patch, an additional interpretation tree branch is added to each node. The general search strategy is based on a depth-first approach in which the search is for the *best* interpretation. However, the search for the best solution can be very time consuming.

There have been many attempts to improve on the basic idea of the Grimson Lozano-Perez algorithm—a useful summary may be found in [Fisher, 1994]. Typically, correct interpretations are determined early on, and considerable time is spent by attempts to improve them further. Thus, a **cut-off** threshold is used to discontinue the search for an interpretation when the cut-off threshold is exceeded. This approach was found to be highly

significant in improving the search times without adversely affecting the search results [Grimson and Lozano-Perez, 1987; Grimson, 1990]. [Fisher, 1993] divides a model into a tree of progressively smaller sub-models which are combined to produce an overall match, while [Fletcher et al., 1996] uses a coarse-to-fine approach to describing features of surfaces to be matched (in this case, 3D data derived from MR head scans).

Yet another approach has demonstrated practical applicability for assessing similarity of medical images for database retrieval. Here, Voronoi diagrams representing arrangement of regions in an image (Section 4.2.3) were used together with a tree-based metric representing Voronoi diagram similarity [Tagare et al., 1995]. The approach presented in Section 10.6 is a more recent example of a fast and efficient use of a classifier cascade organized in a decision tree [Viola and Jones, 2001; Viola et al., 2003].

10.8 SEMANTIC IMAGE SEGMENTATION AND UNDERSTANDING

This section presents a higher-level extension of region growing methods which were discussed in Section 6.3. These ideas fall under the heading of this chapter, rather than simple segmentation, for a number of reasons: Semantic approaches represent a significantly advanced field of image segmentation and as such require a well-developed set of techniques not fully covered until this point; while from another viewpoint, semantic segmentation includes image region interpretation and may result in image understanding and therefore should be included in this chapter. Whichever, it is considered appropriate to present semantic segmentation methods at this point, after the reader is comfortable with the necessary background material: region growing, object description, minimum error classification, contextual classification, image understanding strategies, etc.

Algorithms already discussed in Section 6.3 merge regions on the basis of general heuristics using local properties of regions, and may be referred to as syntactic information-based methods. Conversely, semantic information representing higher-level knowledge was included in [Feldman and Yakimovsky, 1974] for the first time. It is intuitively clear that including more information, especially information about assumed region interpretation, can have a beneficial effect on the merging process, and it is also clear that context and criteria for global optimization of region interpretation consistency will also play an important role. Further, the approaches described in this section are meant to serve as examples of incorporating context, semantics, applying relaxation methods to propagate constraints, and to show how the global consistency function may be optimized—for applications, see also [Cabello et al., 1990; Strat and Fischler, 1991].

The first issue in semantic region growing is the representation of image regions and their inter-relationships. The concept of the region adjacency graph, in which nodes represent regions and arcs connect nodes of adjacent regions, was introduced in Section 4.2.3. An artificial region may surround the image in order to treat all regions consistently. A dual graph can be constructed from the region adjacency graph in which nodes correspond to intersecting points of boundary segments of different regions and arcs correspond to boundary segments. An example of a region adjacency graph and its dual is shown in Figure 10.35. Each time two regions are merged, both graphs change— the following algorithm [Ballard and Brown, 1982] describes how to update the region adjacency graph and its dual after merging two regions R_i and R_j.

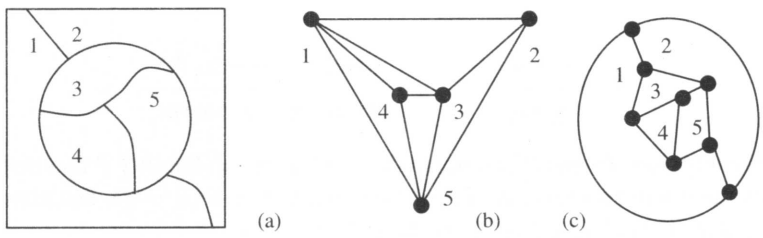


Figure 10.35: Region adjacency graphs. (a) Segmented image. (b) Region adjacency graph. (c) Dual graph.

Algorithm 10.14: Updating a region adjacency graph and dual to merge two regions

1. *Region adjacency graph*
   (a) Add all nonexistent arcs connecting region R_i and all regions adjacent to R_j.
   (b) Remove the node R_j and all its arcs from the graph.
2. *Dual graph*
   (a) Remove all arcs corresponding to the boundaries between regions R_i and R_j from the graph.
   (b) For each node associated with these arcs:
   - If the number of arcs associated with the node is equal to 2, remove this node and combine the arcs into a single one.
   - If the number of arcs associated with the node is larger than 2, update the labels of arcs that corresponded to parts of borders of region R_j to reflect the new region label R_i.

The region adjacency graph is one in which costs are associated with both nodes and arcs, implying that an update of these costs must be included in the given algorithm as node costs change due to the connecting two regions R_i and R_j.

10.8.1 Semantic region growing

A classical method of semantic region growing is now presented [Feldman and Yakimovsky, 1974]. Consider remotely sensed photographs, in which regions can be defined with interpretations such as *field*, *road*, *forest*, *town*, etc. It then makes sense to merge adjacent regions with the same interpretation into a single region. The problem is that the interpretation of regions is not known and the region description may give unreliable interpretations. In such a situation, it is natural to incorporate context into the region merging using a priori knowledge about relations (unary, binary) among adjacent regions, and then to apply constraint propagation to achieve globally optimal segmentation and interpretation throughout the image.

A region merging segmentation scheme is now considered in which semantic information is used in later steps, with the early steps being controlled by general heuristics similar to those given in Section 6.3. Only after the preliminary heuristics have terminated are semantic properties of existing regions evaluated, and further region merging is either allowed or restricted; these are steps 4 and 6 of the next algorithm. The same notation is used as in the previous section: A region R_i has properties X_i, its possible labels are denoted $\theta_i \in \{\omega_1, ..., \omega_R\}$, and $P(\theta_i = \omega_k)$ represents the probability that the interpretation of the region R_i is ω_k.

Algorithm 10.15: Semantic region merging

1. Initialize a segmentation with many small regions.
2. Merge all adjacent regions that have at least one weak edge on their common boundary.
3. For preset constants c_1 and c_2, and threshold T_1, merge neighboring regions R_i and R_j if $S_{ij} \leq T_1$, where

$$S_{ij} = \frac{c_1 + a_{ij}}{c_2 + a_{ij}} \qquad a_{ij} = \frac{(\text{area}_i)^{1/2} + (\text{area}_j)^{1/2}}{\text{perimeter}_i \quad \text{perimeter}_j}. \tag{10.38}$$

4. For all adjacent regions *Ri* and *Rj*, compute the conditional probability P that their mutual border B_{ij} separates them into two regions of the same interpretation ($\theta_i = \theta_j$), equation (10.41). Merge regions R_i and R_j if P is larger than a threshold T_2. If no two regions can be so merged, continue with step 5.

5. For each region R_i, compute the initial conditional probabilities

$$P(\theta_i = \omega_k | X_i)\ k = 1, \ldots, R. \tag{10.39}$$

6. Repeat this step until all regions are labeled as *final*. Find a *non-final* region with the highest confidence C_i in its interpretation [equation (10.43)]; label the region with this interpretation and mark it as *final*. For each *non-final* region R_j and each of its possible interpretations w_k, $k = 1, ..., R$, update the probabilities of its interpretations according to equation (10.44).

The first three steps of the algorithm do not differ in essence from Algorithm 6.18, but the final two steps, where semantic information has been incorporated, are very different and represent a variation of a serial relaxation algorithm combined with a depth-first interpretation tree search. The goal is to maximize an objective function

$$F = \prod_{i,j=1,\ldots,R} P\left(B_{ij} \text{ is between } \theta_i, \theta_j | X(B_{ij})\right) \prod_{i=1,\ldots,R} P(\theta_i | X_i) \prod_{j=1,\ldots,R} P(\theta_j | X_j) \tag{10.40}$$

for a given image partition.

The probability that a border B_{ij} between two regions R_i and R_j is a false one must be found in step 4. This probability P can be found as a ratio of conditional probabilities; let P_t denote the probability that the boundary should remain, and P_f denote the probability that the boundary is false (i.e., should be removed and the regions should be merged), and $X(B_{ij})$ denote properties of the boundary B_{ij}: Then

$$P = \frac{P_f}{P_t + P_f}, \tag{10.41}$$

where

$$P_f = \sum_{k=1}^{R} P[\theta_i = \theta_j | X(B_{ij})]\ P(\theta_i = \omega_k | X_i)\ P(\theta_j = \omega_k | X_j),$$

$$P_t = \sum_{k=1}^{R} \sum_{l=1;k\neq l}^{R} P[\theta_i = \omega_k \text{ and } \theta_j = \omega_l | X(B_{ij})]\ P(\theta_i = \omega_k | X_i)\ P(\theta_j = \omega_l | X_j). \tag{10.42}$$

The confidence C_i of interpretation of the region R_i (step 6) can be found as follows. Let θ_i^1, θ_i^2 represent the two most probable interpretations of region R_i. Then

$$C_i = \frac{P\left(\theta_i^1 | X_i\right)}{P\left(\theta_i^2 | X_i\right)}. \tag{10.43}$$

After assigning the final interpretation θ_f to a region R_f, interpretation probabilities of all its neighbors R_j (with *non-final* labels) are updated to maximize the objective function (10.40):

$$P_{\text{new}}(\theta_j) = P_{\text{old}}(\theta_j)\ P(B_{fj} \text{ is between regions labeled } \theta_f, \theta_j | X(B_{fj})). \tag{10.44}$$

The computation of these conditional probabilities is very expensive in terms of time and memory. It may be advantageous to compute them beforehand and refer to table values during processing; this table must have been constructed with suitable sampling.

It should be understood that appropriate models of the inter-relationship between region interpretations, the collection of conditional probabilities, and methods of confidence evaluation must be specified to implement this approach.

10.8.2 Genetic image interpretation

The previous section described the first historical semantic region growing method, which is still conceptually up to date. However, there is a fundamental problem in the region growing segmentation approach—the results are sensitive to the split/merge order (see Section 6.3). The conventional split-and-merge approach usually results in an under-segmented or an over-segmented image. It is practically impossible to stop the region growing process with a high confidence that there are neither too many nor too few regions in the image.

A method [Pavlidis and Liow, 1990] was mentioned in Section 6.3.3 in which region growing always resulted in an over-segmented image and post-processing steps were used to remove false boundaries. A similar approach of removing false over-segmented regions can be found in a conceptually very different knowledge-based morphological region growing algorithm based on watersheds for graphs [Vincent and Soille, 1991]. Further, conventional region growing approaches are based on evaluation of homogeneity criteria and the goal is either to split a non-homogeneous region or to merge two regions, which may form a homogeneous region. Remember that the result is sensitive to the merging order; therefore, even if a merge results in a homogeneous region, it may not be optimal. In addition, there is no mechanism for seeking the optimal merges. Consequently, the semantic region growing approach to segmentation and interpretation starts with an over-segmented image in which some merges were not best possible. The semantic process is then trying to locate the maximum of some objective function by grouping regions which may already be incorrect and is therefore trying to obtain an optimal image interpretation from partially processed data where some significant information has already been lost. Further, conventional semantic region growing merges regions in an interpretation level only and does not evaluate properties of newly merged regions. It also very often ends in a local optimum of region labeling; the global optimum is not found because of the character of the optimization. Unreliable image segmentation and interpretation of complex images results. The genetic image interpretation method solves these basic problems in the following manner.

- Both region merging and splitting is allowed; no merge or split is ever final, a better segmentation is looked for even if the current segmentation is already good.
- Semantics and higher-level knowledge are incorporated into the main segmentation process, not applied as post-processing after the main segmentation steps are over.
- Semantics are included in an objective evaluation function (that is similar to conventional semantic-based segmentation).
- In contrast to conventional semantic region growing, any merged region is considered a contiguous region in the semantic objective function evaluation, and all its properties are measured.
- The genetic image interpretation method does not look for local maxima; its search is likely to yield an image segmentation and interpretation specified by a (near) global maximum of an objective function.

The genetic image interpretation method is based on a **hypothesize and verify** principle. An objective function (similar to the objective functions used in previous sections) which evaluates the quality of a segmentation and interpretation is optimized by a genetic algorithm (the basics of which were presented in Section 9.6.1). The method is initialized with an over-segmented image called a **primary segmentation**, in which starting regions are called **primary regions**. Primary regions are repeatedly merged into current regions during the segmentation process. The genetic algorithm is responsible for generating new populations of feasible image segmentation and interpretation hypotheses.

An important property of genetic algorithms is that the whole population of segmentations is tested in a single processing step, in which better segmentations survive and others die (see Section 9.6.1). If the objective function suggests that some merge of image regions was a good merge, it is allowed to survive into the next generation of image segmentation (the code string describing that particular segmentation survives), while bad region merges are removed (their description code strings die).

The **primary region adjacency graph** is the adjacency graph describing the primary image segmentation. The **specific region adjacency graph** represents an image after the merging of all adjacent regions of the same interpretation into a single region (collapsing the primary region adjacency graph). The genetic

algorithm requires any member of the processed population to be represented by a code string. Each primary region corresponds to one element in the code string; this correspondence is made once at the beginning of the segmentation/interpretation process. A region interpretation is given by the current code string in which each primary region of the image corresponds uniquely to some specific position. Each feasible image segmentation defined by a generated code string (segmentation hypothesis) corresponds to a unique specific region adjacency graph. The specific region adjacency graphs serve as tools for evaluating objective segmentation functions. The specific region adjacency graph for each segmentation is constructed by collapsing a primary region adjacency graph.

Design of a segmentation optimization function (the fitness function in genetic algorithms) is crucial for a successful image segmentation. The genetic algorithm is responsible for finding an optimum of the objective function. Nevertheless, the optimization function must really represent segmentation optimality. To achieve this, the function must be based on properties of image regions and on relations between the regions—a priori knowledge about the desired segmentation must be included in the optimization criterion.

An applicable objective function may be similar to that given in equation (10.36), keeping in mind that the number of regions N is not constant since it depends on the segmentation hypothesis.

The conventional approach evaluates image segmentation and interpretation confidences of all possible region interpretations. Based on the region interpretations and their confidences, the confidences of neighboring interpretations are updated, some being supported and others becoming less probable. This conventional method can easily end at a consistent but sub-optimal image segmentation and interpretation. In the genetic approach, the algorithm is fully responsible for generating new and increasingly better hypotheses about image segmentation. Only these hypothetical segmentations are evaluated by the objective function (based on a corresponding specific region adjacency graph). Another significant difference is in the region property computation—as mentioned earlier, a region consisting of several primary regions is treated as a single region in the property computation process which gives a more appropriate region description.

Optimization criteria consist of three parts. Using the same notation as earlier, the objective function consists of:

- A confidence in the interpretation θ_i of the region R_i according to the region properties X_i

$$C(\theta_i|X_i) = P(\theta_i|X_i)\,. \tag{10.45}$$

- A confidence in the interpretation θ_i of a region R_i according to the interpretations θ_j of its neighbors R_j

$$C(\theta_i) = \frac{C(\theta_i|X_i)\sum_{j=1}^{N_A}[r(\theta_i,\theta_j)\,C(\theta_i|X_j)]}{N_A}\,, \tag{10.46}$$

  where $r(\theta_i, \theta_j)$ represents the value of a compatibility function of two adjacent objects R_i and R_j with labels θ_i and θ_j, N_A is the number of regions adjacent to the region R_i (confidences C replace the probabilities P used in previous sections because they do not satisfy necessary conditions which must hold for probabilities; however, the intuitive meaning of interpretation confidences and interpretation probabilities remains unchanged).

- An evaluation of interpretation confidences in the whole image

$$C_{\text{image}} = \frac{\sum_{i=1}^{N_R} C(\theta_i)}{N_R} \tag{10.47}$$

  or

$$C'_{\text{image}} = \sum_{i=1}^{N_R}\left(\frac{C(\theta_i)}{N_R}\right)^2\,, \tag{10.48}$$

  where $C(\theta_i)$ is computed from equation (10.46) and N_R is the number of regions in the corresponding specific region adjacency graph.

The genetic algorithm attempts to optimize the objective function C_{image}, which represents the confidence in the current segmentation and interpretation hypothesis.

As presented, the segmentation optimization function is based on both unary properties of hypothesized regions and on binary relations between these regions and their interpretations. A priori knowledge about the characteristics of processed images is used in evaluation of the local region confidences $C(\theta_i|X_i)$, and the compatibility function $r(\theta_i, \theta_j)$ represents the confidence that two regions with their interpretations can be present in an image in the existing configuration.

The method is described by the following algorithm.

Algorithm 10.16: Genetic image segmentation and interpretation

1. Initialize the segmentation into primary regions, and define a correspondence between each region and the related position of its label in the code strings generated by a genetic algorithm.
2. Construct a primary region adjacency graph.
3. Pick the starting population of code strings at random. If a priori information is available that can help to define the starting population, use it.
4. *Genetic optimization.* Collapse a region adjacency graph for each code string of the current population (Algorithm 10.14). Using the current region adjacency graphs, compute the value of the optimization segmentation function for each code string from the population.
5. If the maximum of the optimization criterion does not increase significantly in several consecutive steps, go to step 7.
6. Let the genetic algorithm generate a new population of segmentation and interpretation hypotheses. Go to step 4.
7. The code string with the maximum confidence (the best segmentation hypothesis) represents the final image segmentation and interpretation.

A simple example

Consider an image of a ball on a lawn (see Figure 10.36). Let the interpretation labeling be B for *ball* and L for *lawn*, and let the following higher-level knowledge be included: *There is a circular ball in the image* and *the ball is inside the green lawn region.* In reality some more a priori knowledge would be added even in this simple example, but this knowledge will be sufficient for our purposes. The knowledge must be stored in appropriate data structures.

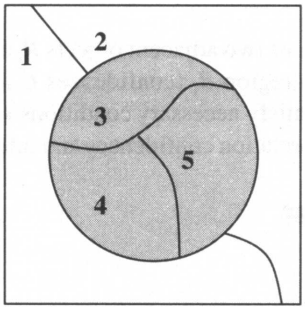


Figure 10.36: A simulated scene 'ball on the lawn'.

- Unary condition: Let the confidence that a region is a *ball* be based on its compactness (see Section 8.3.1),

$$C(\theta_i = B|X_i) = \text{compactness } (R_i) \tag{10.49}$$

and let the confidence that a region is *lawn* be based on its greenness,

$$C(\theta_i = L|X_i) = \text{greenness } (R_i)\ . \tag{10.50}$$

Let the confidences for regions forming a perfect ball and perfect lawn be equal to one

$$C(B|\text{circular}) = 1 \qquad C(L|\text{green}) = 1 .$$

- Binary condition: Let the confidence that one region is positioned inside the other be given by a compatibility function

$$r(B \text{ is inside } L) = 1 \tag{10.51}$$

and let the confidences of all other positional combinations be equal to zero.

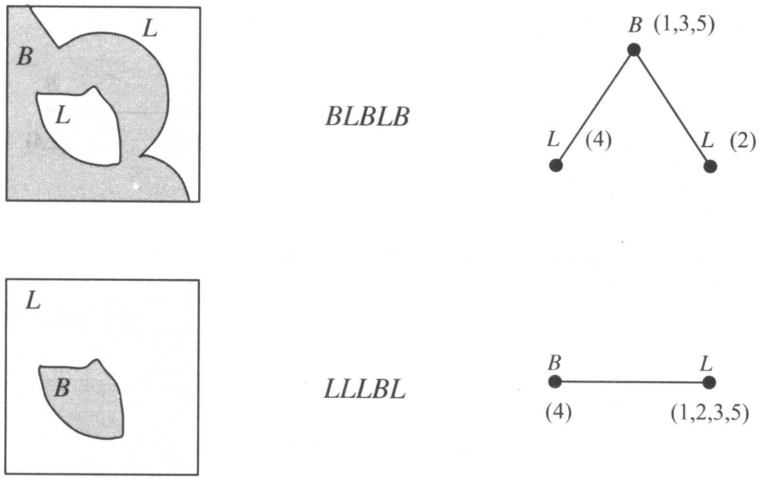


Figure 10.37: Starting hypotheses about segmentation and interpretation: interpretation, corresponding code strings, and corresponding region adjacency graphs.

The unary condition says that the more compact a region is, the better its circularity, and the higher the confidence that its interpretation is a ball. The binary condition is very strict and claims that a ball can only be completely surrounded by a lawn.

Suppose the primary image segmentation consists of five primary regions $R_1, \ldots, R_5$ (see Figure 10.36); the primary region adjacency graph and its dual are in shown in Figure 10.35. Let the region numbers correspond to the position of region labels in code strings which are generated by the genetic algorithm as segmentation hypotheses and assume, for simplicity, that the starting population of segmentation hypotheses consists of just two strings (in any practical application the starting population would be significantly larger). Let the starting population be picked at random:

BLBLB
LLLBL

—this represents segmentation hypotheses as shown in Figure 10.37. After a random crossover between second and third positions, the population is as follows; confidences reflect the circularity of the region labeled *ball* and the positioning of the region labeled *ball* inside the *lawn* region—the confidence computation is based on equation (10.47):

$$BL|BLB\ C_{\text{image}} = 0.00$$
$$LL|LBL\ C_{\text{image}} = 0.12$$
$$LLBLB\ C_{\text{image}} = 0.20$$
$$BLLBL\ C_{\text{image}} = 0.00$$

The second and the third segmentation hypotheses are the best ones, so they are reproduced and another crossover is applied; the first and the fourth code strings die (see Figure 10.38):

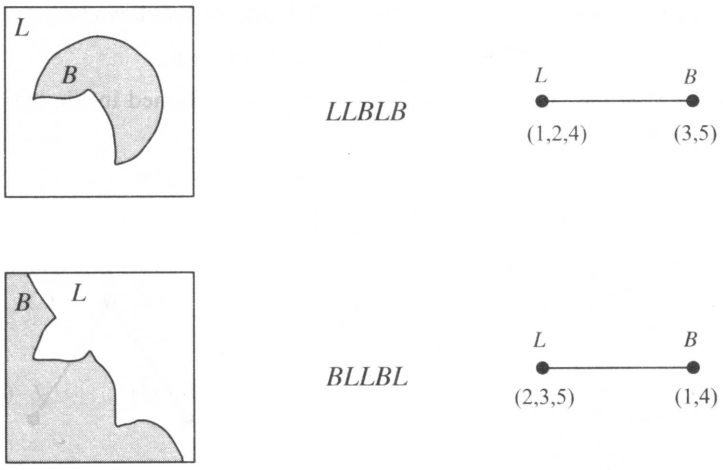


Figure 10.38: Hypotheses about segmentation and interpretation: interpretations, corresponding code strings, and corresponding region adjacency graphs.

$$LLL|BL\ C_{\text{image}} = 0.12$$
$$LLB|LB\ C_{\text{image}} = 0.20$$
$$LLLLB\ C_{\text{image}} = 0.14$$
$$LLBBL\ C_{\text{image}} = 0.18$$

After one more crossover,

$$LLBL|B\ C_{\text{image}} = 0.20$$
$$LLBB|L\ C_{\text{image}} = 0.18$$
$$LLBLL\ C_{\text{image}} = 0.10$$
$$LLBBB\ C_{\text{image}} = 1.00$$

The code string (segmentation hypothesis) *LLBBB* has a high (the highest achievable) confidence. If the genetic algorithm continues generating hypotheses, the confidence of the best hypothesis will not be any better and so it stops. The optimum segmentation/interpretation is shown in Figure 10.39.

Figure 10.39: Optimal segmentation and interpretation: interpretation, corresponding code string, and region adjacency graph.

Brain segmentation example

The previous example illustrated only the basic principles of the method. Practical applications require more complex a priori knowledge, the genetic algorithm has to work with larger string populations, the primary image segmentation has more regions, and the optimum solution is not found in three steps. Nevertheless, the principles remain the same as was demonstrated when the method is applied to more complex problems, and interpretation of human magnetic resonance brain images [Sonka et al., 1996] is given here as such a complex example.

The genetic image interpretation method was trained on two-dimensional MR images depicting anatomically corresponding slices of the human brain. Knowledge about the unary properties of the specified neuroanatomic structures and about the binary properties between the structure pairs was acquired from manually traced contours in a training set of brain images (Figure 10.41a).

As has been apparent from the definition of the global objective function C_{image} [equation (10.47)], the unary properties of individual regions, hypothesized interpretations of the regions, and binary relationships between regions contribute to the computation of the confidence C_{image}.

In our case, the unary region confidences $C(\theta_i|X_i)$ and the compatibility functions $r(\theta_i, \theta_j)$ were calculated based on the brain anatomy and MR image acquisition parameters. The following approach to the confidence calculations was used in the brain interpretation task [Sonka et al., 1996]:

Unary confidences: The unary confidence of a region was calculated by matching the region's shape and other characteristic properties with corresponding properties representing the hypothesized interpretation (i.e., matching with the a priori knowledge).

Let the set of properties of region R_i be $X_i = \{x_{i1}, x_{i2}, \ldots, x_{iN}\}$. Matching was done for each characteristic of the region $\{x_{ij}\}$, and the unary confidence $C(\theta_i|X_i)$ was calculated as follows:

$$C(\theta_i|X_i) = P(x_{i1})\, P(x_{i2}) \ldots P(x_{iN})\,. \tag{10.52}$$

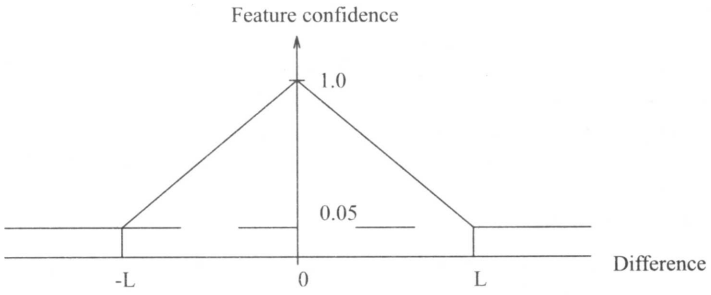


Figure 10.40: Piecewise linear function for calculating unary confidences. L is a limit which depends on the a priori knowledge.

The feature confidences $P(x_{ik})$ were calculated by using the piecewise linear function shown in Figure 10.40. For example, let x_{ik} be the area of region R_i in the specific RAG and let R_i be labeled θ_i. According to a priori knowledge, assume that an object labeled θ_i has an area y_{ik}. Then

$$P(x_{ik}) = \begin{cases} 1.0 - (0.95 \mid x_{ik} - y_{ik} \mid)/L & : \quad \mid x_{ik} - y_{ik} \mid < L\,, \\ 0.05 & : \quad \text{otherwise.} \end{cases}$$

The limit L depends on the strength of the a priori knowledge for each particular feature.

Binary confidences: Binary confidences were defined between two regions based on their interrelationships.

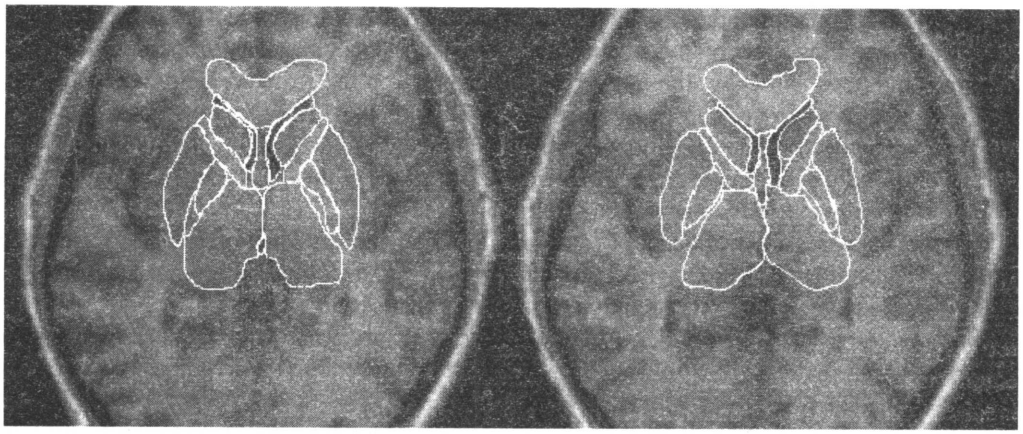
The value of the compatibility function $r(\theta_i, \theta_j)$ was assigned to be in the range [0, 1], depending on the strength of the a priori knowledge about the expected configuration of regions R_i and R_j.

For example, if a region R_i, labeled θ_i, is known always to be inside region R_j, labeled θ_j, then $r(\theta_i$ is_inside $\theta_j) = 1$ and $r(\theta_j$ is_outside $\theta_i) = 1$, whereas $r(\theta_j$ is_inside $\theta_i) = 0$ and $r(\theta_i$ is_outside $\theta_j) = 0$. Thus, low binary confidences serve to penalize infeasible configurations of pairs of regions.

Similarly to the calculation of the unary confidence, the compatibility function was calculated as follows:

$$r(\theta_i, \theta_j) = r(\theta_{ij1})\, r(\theta_{ij2}) \ldots r(\theta_{ijN})\,. \tag{10.53}$$

Here, $r(\theta_{ijk})$ is a binary relation (example: larger than/smaller than) between regions labeled θ_i and θ_j.



(a) (b)

Figure 10.41: Automated segmentation and interpretation of MR brain images. (a) Observer-defined borders of the neuroanatomic structures correspond closely with (b) Computer-defined borders.

After the objective function C_{image} was designed using a number of brain images from the training set, the genetic brain image interpretation method was applied to testing brain images. For illustration, the primary region adjacency graph typically consisted of approximately 400 regions; a population of 20 strings and a mutation rate $\mu = 1/string\_length$ were used during the genetic optimization. The method was applied to a testing set of MR brain images and offered good image interpretation performance (Figure 10.41).

Semantic image understanding

Conventional semantic region growing methods start with a non-semantic phase and use semantic post-processing to assign labels to regions. Based on the segmentation achieved in the region growing phases, the labeling process is trying to find a consistent set of interpretations for regions. The genetic image interpretation approach functions in a quite different way.

First, there are no separate phases. The semantics are incorporated into the segmentation/interpretation process. Second, segmentation hypotheses are generated first, and the optimization function is used only for evaluation of hypotheses. Third, a genetic algorithm is responsible for generating segmentation hypotheses in an efficient way.

The method can be based on any properties of region description and on any relations between regions. The basic idea of generating segmentation hypotheses solves one of the problems of split-and-merge region growing—the sensitivity to the order of region growing. The only way to re-segment an image in a conventional region growing approach if the semantic post-processing does not provide a successful segmentation is to apply feedback control to change region growing parameters in a particular image part. There is no guarantee that a global segmentation optimum will be obtained even after several feedback re-segmentation steps.

In the genetic image interpretation approach, no region merging is ever final. Natural and constant feedback is contained in the genetic interpretation method because it is part of the general genetic algorithm—this gives a good chance that a (near) global optimum segmentation/interpretation will be found in a single processing stage.

Note that throughout this chapter, the methods cannot and do not guarantee a correct segmentation—all the approaches try to achieve optimality according to the chosen optimization function. Therefore, a priori knowledge is essential to designing a good optimization function. A priori knowledge is often included into the optimization function in the form of heuristics, and moreover, may affect the choice of the starting population of segmentation hypotheses, which can affect computational efficiency.

An important property of the presented genetic image understanding method is the possibility of parallel implementation. Similarly to the relaxation algorithm, this method is also naturally parallel. Moreover, there is a straightforward generalization leading to a genetic image segmentation and interpretation in three dimensions. Considering a set of image planes forming a three-dimensional image (such as MR or CT images), a primary segmentation can consist of regions in all image planes and can be represented by a 3D primary relational graph. The interesting option is to look for a global 3D segmentation and interpretation optimum using 3D properties of generated 3D regions in a single complex processing stage. In such an application, the parallel implementation would be a necessity.

10.9 HIDDEN MARKOV MODELS

It is often possible when attempting image understanding to model the patterns being observed as a transitionary system. Sometimes these are transitions in time, but they may also be transitions through another pattern; for example, the patterns of individual characters when connected in particular orders represent another pattern that is a word. If the transitions are well understood, and we know the system state at a certain instant, they can be used to assist in determining the state at a subsequent point. This is a well-known idea, and one of the simplest examples is the **Markov model**.

A Markov model assumes a system may occupy one of a finite number of states $X_1, X_2, X_3, \ldots, X_n$ at times $t_1, t_2, \ldots$, and that the probability of occupying a state is determined solely by recent history. More specifically, a first-order Markov model assumes these probabilities depend only on the preceding state; thus a matrix $A = a_{ij}$ will exist in which

$$a_{ij} = P(\text{system is in state } j \mid \text{system was in state } i)\,. \tag{10.54}$$

Thus $0 \le a_{ij} \le 1$ and $\sum_{j=1}^{n} a_{ij} = 1$ for all $1 \le i \le n$. The important point is that these parameters are time independent—the a_{ij} do not vary with t. A second-order model makes similar assumptions about probabilities depending on the last two states, and the idea generalizes obviously to order-k models for $k = 3, 4, \ldots$

A trivial example might be to model weather forecasting: suppose that the weather on a given day may be *sunny* (1), *cloudy* (2), or *rainy* (3) and that the day's weather depends probabilistically on the preceding day's weather *only*. We might be able to derive a matrix A,

$$A = \begin{array}{c} \\ \text{sun} \\ \text{cloud} \\ \text{rain} \end{array} \begin{array}{c} \begin{array}{ccc} \text{sun} & \text{cloud} & \text{rain} \end{array} \\ \begin{bmatrix} 0.50 & 0.375 & 0.125 \\ 0.25 & 0.125 & 0.625 \\ 0.25 & 0.375 & 0.375 \end{bmatrix} \end{array} \tag{10.55}$$

so the probability of rain after a sunny day is 0.125, the probability of cloud after a rainy day is 0.375, and so on.

In many practical applications, the states are not directly observable, and instead we observe a different set of states $Y_1, \ldots, Y_m$ (where possibly $n \neq m$), where we can only guess the exact state of the system from the probabilities

$$b_{kj} = P(Y_k \text{ observed} \mid \text{system is in state } {}_j)\,,$$

so $0 \le b_{kj} \le 1$ and $\sum_{k=1}^{m} b_{kj} = 1$. The $n \times m$ matrix B that is so defined is also time independent; that is, the observation probabilities do not depend on anything except the current state, and in particular not on how that state was achieved, or when.

Extending the weather example, it is widely believed that the moistness of a piece of seaweed is an indicator of weather; if we conjecture four states, *dry* (1), *dryish* (2), *damp* (3) or *soggy* (4), and that the actual weather is probabilistically connected to the seaweed state, we might derive a matrix such as

$$B = \begin{array}{c} \\ \text{dry} \\ \text{dryish} \\ \text{damp} \\ \text{soggy} \end{array} \begin{array}{c} \begin{array}{ccc} \text{sun} & \text{cloud} & \text{rain} \end{array} \\ \begin{bmatrix} 0.60 & 0.25 & 0.05 \\ 0.20 & 0.25 & 0.10 \\ 0.15 & 0.25 & 0.35 \\ 0.05 & 0.25 & 0.50 \end{bmatrix} \end{array} \tag{10.56}$$

so the probability of observing dry seaweed when the weather is sunny is 0.6, the probability of observing damp seaweed when the weather is cloudy is 0.25, and so on.

A first-order **hidden Markov model** (**HMM**) $\lambda = (\pi, A, B)$ is specified by the matrices A and B together with an n-dimensional vector π to describe the probabilities of the state at time $t = 1$. The time-independent constraints are quite strict and in many cases unrealistic, but HMMs have seen significant practical application. In particular, they are successful in the area of speech processing [Rabiner, 1989], wherein the A matrix might represent the probability of a particular phoneme following another phoneme, and the B matrix refers to a feature measurement of a spoken phoneme (the Fourier spectrum, for example). It is recognized here that the fuzziness of speech means we cannot be certain which feature will be generated by which phoneme. The same ideas have seen wide application in optical character recognition (OCR) (for example, [Agazzi and Kuo, 1993],) and related areas, where the A matrix might refer to letter successor probabilities, and again the B matrix is a probabilistic description of which features are generated by which letters.

A HMM poses three questions:

Evaluation: Given a model and a sequence of observations, what is the probability that the model actually generated those observations? If two different models are available, $\lambda_1 = (\pi_1, A_1, B_1)$ and $\lambda_2 = (\pi_2, A_2, B_2)$, this question indicates which one better describes some given observations. For example, if we have two such models, together with a known weather sequence and a known sequence of seaweed observations, which model, λ_1 or λ_2, is the best description of the data?

Decoding: Given a model $\lambda = (\pi, A, B)$ and a sequence of observations, what is the most likely underlying state sequence? For pattern analysis, this is the most interesting question, since it permits an optimal estimate of what is happening on the basis of a sequence of feature measurements. For example, if we have a model and a sequence of seaweed observations, what is most likely to have been the underlying weather sequence?

Learning: Given knowledge of the set $X_1, X_2, X_3, \ldots, X_n$ and a sequence of observations, what are the best parameters π, A, B if the system is indeed a HMM? For example, given a known weather sequence and a known sequence of seaweed observations, what model parameters best describe them?

HMM Evaluation

To determine the probability that a particular model generated an observed sequence, it is straightforward to evaluate all possible hidden state sequences, calculate their probabilities, and multiply by the probability that the sequence in question generated the observations in hand. If

$$Y^k = (Y_{k_1}, Y_{k_2}, \ldots, Y_{k_T})$$

is a T long sequence of observations, and

$$X^i = (X_{i_1}, X_{i_2}, \ldots, X_{i_T})$$

is a state sequence, we require

$$P(Y^k) = \sum_{X^i} P(Y^k \mid X^i) P(X^i).$$

This quantity is given by summing over all possible sequences X^i, and for each such, determining the probability of the given observations; these probabilities are available from the B matrix, while the transition probabilities of X^i are available from the A matrix. Thus