Włodzimierz Kasprzak

SEMANTIC NETWORK–BASED SCENE ANALYSIS IN RANGE DATA

Warsaw, January 1990

Abstract . Streszczenie

Scene analysis deals with compound objects, which are either man–made or natural. CSG–Models are often provided for the construction of man–made solids in CAD systems. They are seldom used in vision systems because of the main problem of transforming the boundaries detected in the image into primitive volume types and point set operations over volumes, that must be solved during the recognition. In this report a solid model, that directly corresponds to the CSG-modelling is used for object recognition in range– or stereo–image–data. Because of the domain-independent nature of the recognition approach considered, the framework of knowledge-based analysis is applied. It consists of a procedural semantic net (for knowledge representation) with several domain–independent "IF–THEN" rules for instantiation and modification (use of knowledge) and of an A^*–tree search based bi-driven control algorithm (recognition strategy).

Analiza scen w obrazach z mapą głębi w oparciu o sieci semantyczne

Dziedzina *analizy scen* zajmuje się złożonymi obiektami o charakterze naturalnym bądz sztucznie stworzonymi przez człowieka. Modele konstruktywnej geometrii brył (ang. CSG) są często stosowane w systemach wspomagania projektanta (ang. CAD) do tworzenie sztucznych brył. Nie stosuje się ich w systemach *komputerowej wizji* z uwagi na zasadniczą trudnośc przekształcenia w trakcie procesu rozpoznawania brzegów odkrytych w obrazie w elementarne objętości i operacje zbiorów nad nimi. W niniejszym raporcie dla rozpoznawania obiektów w obrazach z mapą głębi (lub w stereo–parach obrazów) stosuje się model bryłowy, który bezposrednio koresponduje ze schematem CSG. Z uwagi na niezależnośc rozpatrywanego podejścia do rozpoznawania od konkretnej dziedziny, można zastosowac schemat analizy opartej o *bazę wiedzy*. Schemat ten składa się z reprezentacji wiedzy w postaci proceduralnych sieci, z kilku niezależnych od dziedziny reguł tworzenia instantów i modyfikowanych konceptów (o formacie "JESLI – TO") i z dwu–kierunkowej strategii sterowania bazującej na algorytmie optymalnego przeszukiwania drzewa – A^*.

Chapter 1

Introduction

One of the most important problems in **computer vision** [1] is the recognition of 3–dimensional objects in multi–object scenes on base of their partial views. There are two main approaches to **3–D object recognition** [2–4]:
1) the *prototype–free* partitioning of the scene into objects and
2) the *model–based* recognition of objects.
In the first approach we recognize objects without using explicit models of objects, i.e. we partition regions with surrounding lines and vertices into bodies and classify them. This is a data–driven analysis in which a *domain–dependent* knowledge is applied. In the second approach we try to detect and to loalize objects by *matching* the *image primitives* with model elements.

In natural or industrial scenes we deal with **compound objects** (with complex parts), which are either *man–made* or *natural* (Fig. 1.1). For their recognition we need a scheme for the *generation* of object classes and additional *analysis* knowledge.

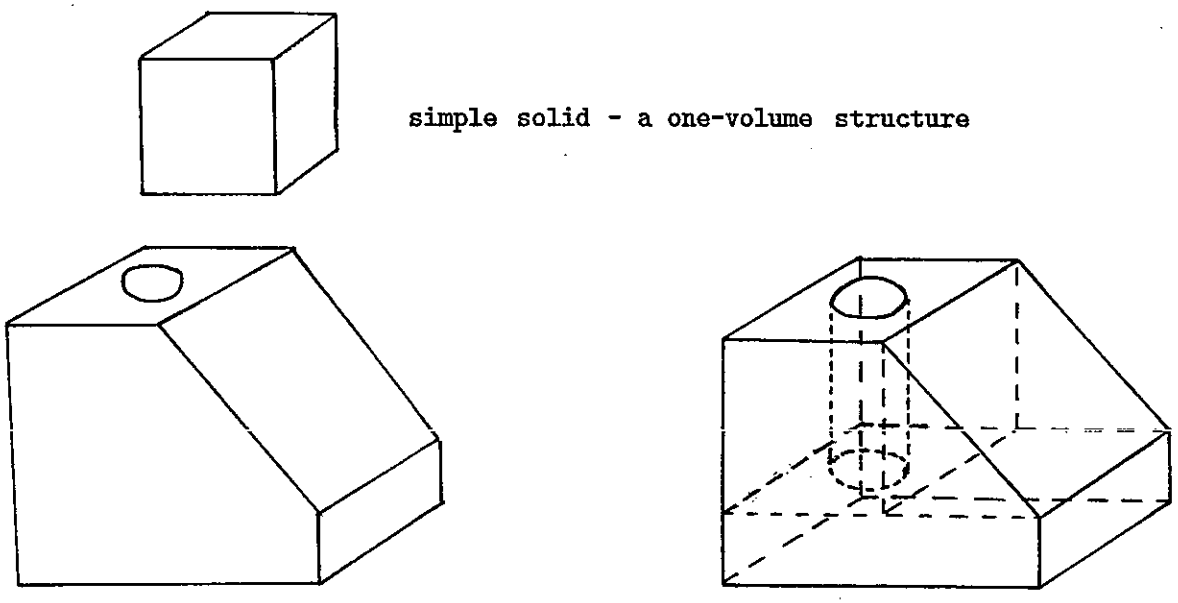


Figure 1.1: Example of a compound solid

A proper representation scheme for natural objects is not available so far. At other hand CAD–Models are provided for constructing man-made objects. In the most popular modelling scheme – *constructive solid geometry* (CSG) a compound solid class is defined as an ordered set of simple volumes and

Boolean volumetric operations over them. Only the primitive volumes may be defined by faces and edges (*B–representation*) or by sweep and rotation operations for point sets (Fig. 1.2).

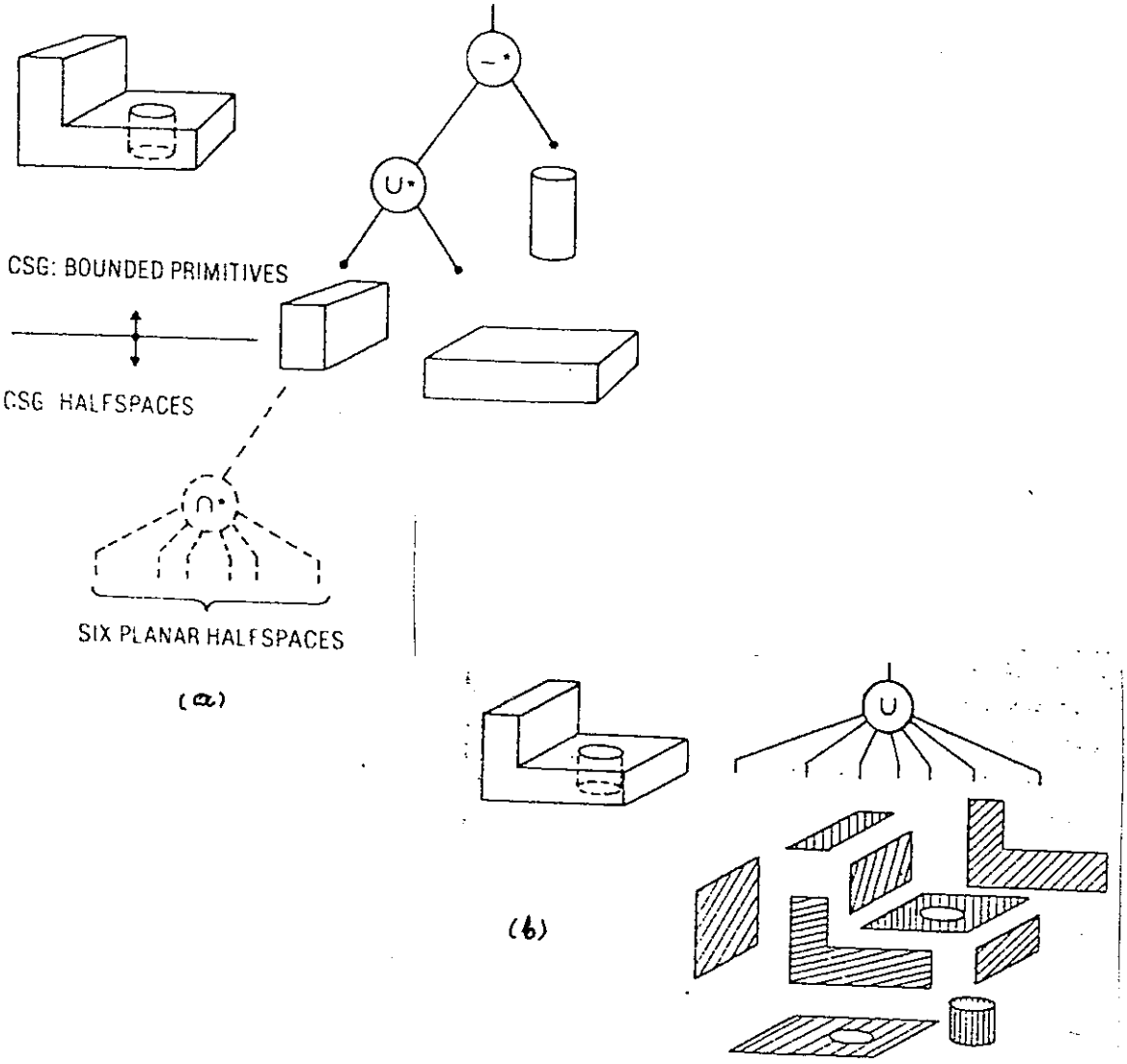


Figure 1.2: Main CAD–modelling schemas: (a) CSG , (b) B–Rep.

In this report a *model–based recognition* of CSG–defined objects is described. This is a challenging problem, because following distortions of surfaces visible in the image can exist, due to the Boolean operations or due to occlusions :
1) some faces and edges between primitive shells of a solid are partially or fully removed
2) a compound solid may contain negative faces.

Because of the *domain–independent* nature of considered recognition approach, the framework of **knowledge–based analysis** [5–7] can be applied to it. For this purpose **procedural semantic nets** with several domain–independent "IF–THEN" inference rules are used and are utilized by an optimal tree search algorithm.

We focus on the recognition of objects given in **range–images** (or **stereo–pairs** of images).

Section 2 is a short introduction to knowledge representation methods and their control techniques, and a reviev of model–based vision systems is given also. In section 3 the problem of range image segmentation and 3–D object recognition is outlined.
In section 4 the structure of assumed vision system and the object recognition approach are outlined. A short introduction to a semantic net implementation ERNEST [8, 9] is included. Extensions are proposed, that concern the integration of *data–driven* elements into the *model–to–image matching* mechanism used in ERNEST. Two particular features of scene analysis considered here are:
– MULTIPLE instances of one concept (i.e. a varying number of objects in a scene)
– the imperfect and incomplete segmentation data requires a verification of generated hypotheses by testing GLOBAL visibility relations
The semantic net–based approach to the recognition of CSG–defined objects in multi–object scenes is described in more details in sections 5 and 6. Section 5 is devoted to the principal construction of the *vision model*, whereas in section 6 the general *bi–driven control* algorithm is given.

List of Contents:

Chapter 2

Knowledge–based scene analysis

Three **processing levels** are generally distinguished in a computer vision system:

- **low–level** (array processing level) – The sensor data is processed in order to achieve a low–level symbolic image description in terms of *image primitives.*
- **intermediate level** (object recognition level) – Its goal is a scene description in terms of *objects.*
- **high–level** (image understanding, cognitive level) – Its goal is a human–like scene *interpretation* and activation of *actions.*

We consider the problem of model–based object recognition from the point of view of **knowledge–based signal analysis.** Five main *system modules* exist in a knowledge–based system [6, 7]:

1. **Knowledge Base** – Both *declarative* and *procedural* knowledge for domain–independent processing of low–level image descriptions. It consists of the *Model (Facts)* utilized during the analysis and of the *problem solving–knowledge (Model–scheme)* utilized during model acquisition.
2. **Control** – A problem–independent processing strategy, which activates the appropriate *inference* processes with relevant subsets of the available data. Additional processes under control can be *dialog* and *explanation* components. We distinguish two control components: 1) the *recognition strategy* (on the object–recognition level) and 2) the *understanding strategy* (on the high level of processing).
3. **Data Base** – Representation of temporary results and of image descriptions.
4. **Methods** – Domain–dependent mainly procedural knowledge for low–level processing.
5. **Knowledge Acquisition** – Structuring and incorporation of new elements into the Model.

2.1 Knowledge representation, inference and control methods

The main knowledge representation methods, that were proposed for signal understanding systems are as follows:

- **Predicate logic**
  The syntax of the first order predicate language consists of basic *items*, *atoms* over items and of *well–formed formulas*, that are sets of atoms combined by *logical operators* and *quantifiers.* The semantics is given by *interpretations* of the basic items.
  A *clause–form* syntax is often used. A *clause* is an ordered pair of sets of atoms: $A_1, ..., A_n - - > B_1, ..., B_m$. A semantics of such an expression with variables $x_1, ..., x_k$ is equivalent to the semantics of a formula: $x_1, ..., x_k(A_1 \wedge ... \wedge A_n) => (B_1 \vee ... \vee B_m)$.
  A formula (clause) C *logically follows* from a set of formulas (clauses) if every interpretation that makes S true also makes C true. In nonclausal predicate logic, inferences are rewritings of axioms

and previously established formulae in accordance with *rules of inference* such as *modus ponens, modus tollens, substitution.* Goal of the clausal form analysis is to show the unsatisfiability of S and C taken together, by the use of the *resolution* rule: "matching a condition of one clause A with a conclusion of another clause B. The derived clause, called the *resolvent*, consists of the unmatched conditions and conclusions of A and B instantiated by the matching substitution".

- **Production rules**
  The knowledge base consists of production rules in the form: *IF (condition) THEN (conclusion).* If the left–side *condition* is satisfied, the right–hand *conclusion* can be taken. Thus the use of a production rule is as follows: try to match the condition with the data base, and in case of success perform the action specified by the conclusion.
  Sometimes the selection of rules is integrated into the production rule. In this case they have the following form: *IF (condition) THEN (conclusion) TRUE (next rule(s) A) FALSE (next rule(s) B).*

- **Grammars**

- **Structural approaches**

  Relation structures [10]

  Hierarchical graphs [11 – 13]

  Graph and relation structure grammars – In a previous paper [14] the author has proposed a paradigm for the recognition of 3–D objects in range data in terms of a two–step parsing of *attributed* relation structure grammars. The knowledge is represented by attributed productions. The procedural attachment is provided by attributes of nodes, predicates of edges and by substitution rules associated with each production.
  The first step of the control strategy is a bottom–up hierarchical constraint reduction process, which applies the so called *dominating* parts of the productions in a reversive manner and which produces a layered hypothesis net with object instances being on the top level. The second step is a cycle which consists of: 1) a selection of the best hypothetical scene description; 2) a top–down parsing during which the selected description is refined and verified; 3) in case of failure the inconsistent description is reduced to potentially consistent subdescriptions.

  Frames [15]

  Semantic networks – A *semantic network* [16] is a labeled graph consisting of nodes and links being restricted to some types only. An implementation of a semantic net express the knowledge in terms of *concepts*, their *attributes* (properties), and the hierarchical sub/superclass relationship (*is-a* or *instance–of*) between concepts. A procedural component is usually attached for expressing the semantics of representation. This characterization captures the basic organizational principle underlying frame–based representation languages such as FRL, KRL and KL–ONE.
  Two forms of *inference* are often realized in semantic networks [17]:
  – *inheritance of conceptual attributes*
  – *concept recognition.*
  Inheritance leads an agent to infer properties of a concept based on the properties of its "is a"–ancestor. *Concept recognition* may be described as follows: given a description consisting of a set of properties, find a concept that best matches this description. Thus recognition and inheritance are dual problems. While inheritance seeks a property value of a given concept, recognition seeks a concept that has some specified property values.

The control methods applied in above representations are:

- **Logical inference systems**
  The analysis in predicate logic is *theorem proving*. It tries to deduce a goal formula from the set of *axioms* and actual *facts* by using inference rules.

- **Search techniques**
  A control strategy of a rule–based deduction system can be expressed in terms of a *state–space* search (or graph search). The application of a rule results in the addition of a node to the predecessor node in the search space. The *root* node models the start situation (with the initial set of facts), the *terminal* nodes are the goal nodes (descriptions of initial facts). There are various control strategies for traversion of the state space, e.g. *forward reasoning*, *backward reasoning*, *hypothesize–and–test*, *generate–and–test* (Fig. 2.1).
  Decomposable problems can be solved by the *AND–OR graph* search techniques.

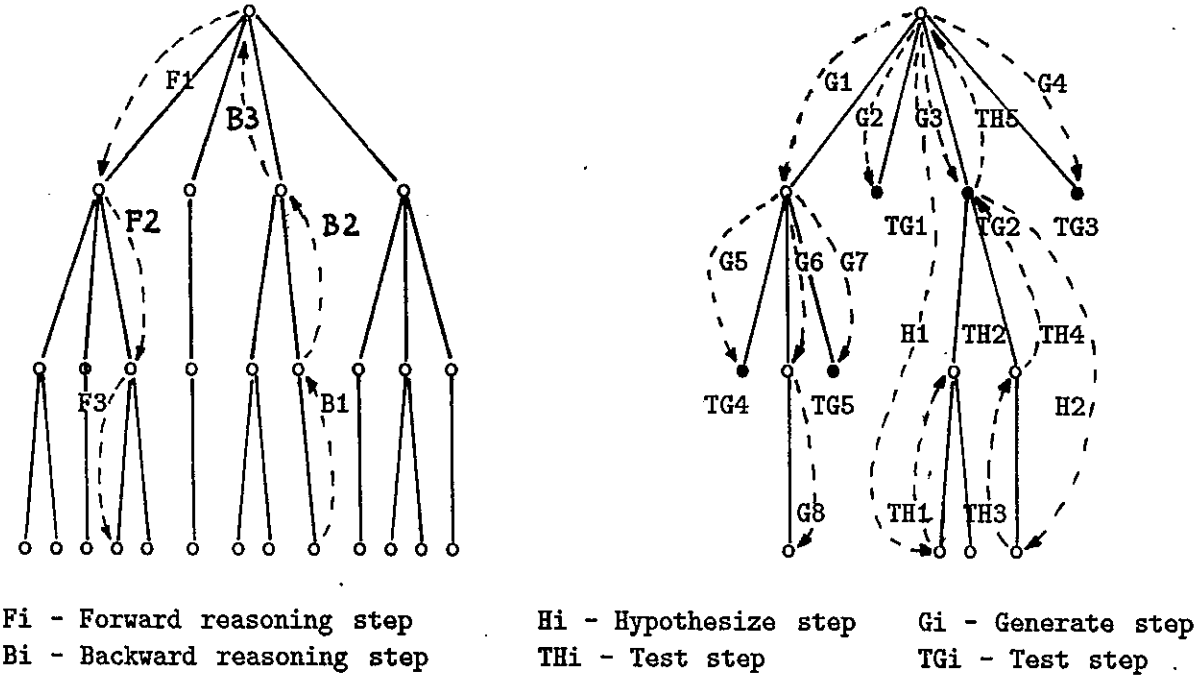


Figure 2.1: Main control strategies for state space search

- **Linguistic analysis**
- **Graph– and relational structure–matching**
- **Relaxation**

2.2 Model–based scene analysis systems

In this report we restrict our interest to the construction of three modules – knowledge base, data base and control, on the object recognition level of processing. The system design is influenced by previously proposed 3–D model–based vision systems for intensity [18–23] and range–image analysis [24]. A short summary of them follows below.

ACRONYM

The *ACRONYM* system from Brooks [18] provides a volumetric object model based on *generalized cones.* The model is abstractly represented by a set of three graphs:

- the *object graph* – The nodes represent object classes, the arcs – *subpart* relations, and *affixment relations* (the relations between the local coordinate system of each node and the object coordinate system). Objects and cameras are placed in the world by constraining the transforms between their coordinate systems and the world coordinate system.
- the *restriction graph* – It represents the constraints of one object class – the nodes are subclasses and the links are subclass–to–class inclusion relations.
- the *prediction graph* – It represents the invariant and quasi–invariant observable image features (primitives) of objects, and relationships between features: *must be, should be, exclusive.*

The knowledge graphs are implemented by a set of *frame* structures. Each particular data structure is an instance of a *unit* and contains *slots* with *fillers.* Because there are various types of units possible, a units's class is given by its slot *class.*

The *recognition strategy* is expressed by an iterated cycle of:

- *search* in the restriction graphs (incremental exploration of the model)
- *prediction* of 2–D object invariants and quasi–invariants : (a) is first used to build the *prediction* graph, which provides a description of features (so called *ribbons* and *ellipses*) and their relations; (b) it also provides inctructions on how to use image feature measurements to constrain the three dimensional models, identifying class memberships and specific 3–D spatial relations.
- prediction–directed *description* of image
- interpretation : (a) local *matching* of predicted features with image features – the measurements are used to put constraints on parameters of predicted objects (i.e. it constrain camera parameters, object size and structure or relations between camera parameters and object size; (b) upward combination of local matches into *clusters* (more global interpretations) – all constraints implied by local matches must be consistent.

The prediction and interpretation processes are expressed by a *production rule system.* The heart of computation is a nonlinear constraint manipulation system (CMS), which is used to determine if the downward propagated constraints match the upward propagated one.

VISIONS

The *VISIONS* system from Hanson & Risemann [19, 20] is a general methodology for knowledge–based vision systems.

Knowledge is represented in a semantic network with nodes called *schemas.* Each schema defines a structured collection of elements in a scene or object and encodes the knowledge of how to recognize the object/scene. A schema has both a *declarative* description, appropriate to the level of detail describing relations between the parts of the schema, and a *procedural* component describing object recognition techniques expressed as a set of hypothesis and verification processes.
Both knowledge base and data base are multilevel representations of two–dimensional symbolic tokens of points, lines and regions, and three–dimensional tokens of surfaces and volumes, and the abstract semantic tokens of object and scenes. The hierarchy is organized via the PART–OF and IS–A relations.

The *control* is structured into a three layer–hierarchy of *model builders*, that incrementally build (*focus, extend* and *verify*) the partial scene interpretations. First a node in the interpretation space is selected, then the level in the knowledge hierarchy and at end a particular schema on this level.

Because the interpretation process is distributed over schemas they communicate using a *blackboard system*, i.e. the results are written to the STM and can be accessed by other processes.

3–D MOSAIC

The *3–D MOSAIC* system from Herman, Kanade & Kuroe [21] incrementally generates a 3–D model of a complex scene from multiple intensity images (views), while assuming enormous scene domain restrictions. The initial model is constructed from the first view. As each successive view is processed, the model is incrementally updated and becomes more accurate and complete.

The scene model is a surface–based description, represented as a graph in terms of symbolic primitives and their topology and geometry.

The incremental construction of the model proceeds as follows:

1. Constructing a wire–frame description – Each view of the scene undergoes analysis which results in a 3–D frame description corresponding to portions of boundaries of objects in the scene. All of its topological elements are tagged as *confirmed*. The first view analysis forms the *initial* state of the *scene model*.

2. Updating the model – The new wire–frame description is *matched* and *merged* with the current model.

3. Surface–based completion – The partial model is converted into a full surface–based description, by predicting new vertices, edges and faces. A task–specific knowledge about available objects is exploited. All elements of the model that were not present in the preceeding views are predicted and tagged as unconfirmed.

Lee & Fu

Lee & Fu [22] propose a vision system architecture with verification feedback. There are three basic recognition processes: (a) object description generation (hypothesis process), (b) model retrieval (prediction), (c) model verification (verification of hypotheses).
They list a group of 2–D primitives and regularity constraints, that relate these primitives with 3–D loops and edges. Additional so called least-slant-angle preference rule is used to compute the 3–D surface orientation of a single region. This partial hypothetical interpretation is propagated to neighboring regions to yield a rough object description in terms of visible surface orientation.

SIGMA

Hwang, Davis & Matsuyama [23] developed a general, domain–independent control strategy, that integrates both *bottom–up* and *top–down* analysis into a single reasoning process. They use a combined *frame/production rule*–representation of the scene model.

The presented framework is based on four reasoning steps. Hypotheses are regarded as *predictions* of the occurencies of objects in the image. Related hypotheses are clustered together (*integration*). A "composite" hypothesis is computed for each cluster (*abstraction*). A hypothesis is *verified* by computing values for those attributes, which are not completely constrained.

The authors constructed an image understanding system called *SIGMA*, which is structured into three processing modules: Low Level (LLVS), High Level (HLVS) and Query Answering Module (QAM). The HLVS module uses object models to interpret the segmentation results and it outputs an *interpretation network*. HLVS performs an iterative reasoning, based on the hypotheses generation–integration–verification principle. At the end of each iteration, the QAM is activated to check whether

the "goal" is accomplished. QAM matches the goal, represented as a query, with the interpretations already constructed.

The model is abstractly a graph with nodes representing objects and two types of arcs *links* and *rules*, representing the relations between these objects.
Basic entities of the knowledge representation are *frames*, which represent the objects with their relations. Each frame may have many associated property lists defined by *slots*.
The knowledge used to compute values of slots is represented by *rules*. Rules used in this context are of procedural nature. *Links* are the third–type frame elements. They are used to describe the generalization/sp ecialization hierarchy of objects (called AKO, CAN–BE resp.). Part–type relations as well as geometrical relations between objects are represented in a frame by rules.
A rule is composed of three parts:

$$< control - condition >, < hypothesis >, < action >.$$

$< Control - condition >$ is a predicate which indicates when a rule can potentially be applied.
$< Hypothesis >$ specifies the description of a desired object that is created when the $< hypothesis - -condition >$ evaluates to true.
$< Action >$ describes the procedure to be performed if $< hypothesis >$ is verified – adding or deleting facts from the iconic/symbolic data base.

As the frames are prototypes of objects, *instances* of frames are interpretations of the image in terms of the model. An instance can be either in a *verified* or *hypothetical* state. The first case indicates that the appearance of the instance is some already located image structure or is a function of the appearances of verified instances. Every instance has a numerical value which is called the *strength* of the instance and is computed by a procedure from the frame definition. This value is used for the focus of attention mechanism of control.

Kuan & Drazovich

For range images a general system outline was presented by Kuan & Drazovich [24]. They extend the top–down recognition method presented in ACRONYM by a bottom–up cue generation process. A hierarchical volumetric representation including four levels (object, cylinder, surface, edge) is used, whereas the ACRONYM is based on a homogeneous Model.

On each level from top to down three processes are performed in sequence: the image–driven *cue generation*, the model–based *prediction* and the *image feature generation*. The processing cycle on a single level is as follows:

1. Predict the 3–D features on the next level of representation on base of matched 3–D features on the current level.

2. Go to next level and use the predictions as guidance for goal-directed feature extraction.

3. Compare the extracted 3–D features with the model in an image-to-model matching. If the object recognition is not achieved at current processing level then go to process 1. on this level.

Chapter 3

3–D object recognition in range data

3.1 Range image segmentation and low–level description

Goal of the 3–D *segmentation* process is to partition the image into meaningful parts or to extract important elements from it. These parts and elements (called *primitives*) are usually classfied into three categories:

- *regions* (surface patches)
- *line segments* or *edges* (region boundaries)
- *junctions* (edge boundaries)

The segmentation process should be domain independent – specific types of objects and their shapes are not assumed here. The only assumption is that an object surface is partially composed of smooth differentiable surfaces.

Techniques for producing the image primitives are *region-based* (mainly applied to dense range images) and *edge–based* (mainly applied to sparse range data). A region–based approach attempts to group the pixels into regions based on the homogenity or similarity of image properties. An edge-based approach attempts to extract discontinuities in image properties and to form the boundaries of regions.

Goal of the *low–level description* process is to describe the image in terms of instances of primitive classes and their features, and in terms of relations between primitives.

3.1.1 Features and descriptors

Similarity of following *region features* has beeen used (1–3) or proposed (4) for region detection:

1. curvature measures – for curved regions
   (e.g. mean curvature – is the average of the minimum and maximum of principal curvatures at each point; Gaussian curvature – is the product of principal curvatures))
2. surface normals – for planar regions
3. similarity of depth and coordinate positions; adjacency of regions
4. texture (roughness) or fractal dimension

Edge features proposed for boundary detection are:

1. depth discontinuities – between object and background (for so called *step* or *jump* edges)
2. orientation discontinuities – between two planar patches facing diferent directions (for *roof* edges)

3. curvature discontinuities – a smooth joining of two curved patches with different shape (for *smooth* edges)

4. zero crossings of curvature or edges – internal boundaries of surface patches (where the above two change sign)

Region classes are determined due to following *descriptors*:
(a) surface types fitting the detected regions
(b) convex or concave surface type

Edge classes are determined due to following *descriptors*:
(a) types of curves connecting the discontinuity points (the curves of surface intersection)
(b) line junction types – the ways in which edges join at surface intersections of planar regions

3.1.2 Segmentation and description of dense range images

Let us assume, that for scene elements which distance from observer exceeds some threshold, no range values are available. If an associated intensity image is provided, those pixels will be labeled as *outdoor* pixels and will underly a 2–D analysis, whereas for the remaining *indoor* pixels a 3–D Model–based analysis can be started.

A. Typical segmentation process

A typical segmentation and description process for dense range data consists of following steps:

1. *Region–features extraction*
   By examining a small (3x3 or 5x5) window of pixels centered at each range pixel, region features can be computed. Appropriate plane normals, gradients- or curvature-maps are obtained out from the range image.

2. *Labeling (or classification) of pixels*
   Based on above features the pixels are classified into surface or boundary elements. A labeled (or classified) image is obtained.

3. *Extraction of boundary features*
   Boundary pixels are detected in the set of classfied pixels (roof, smooth edges) or in the initial range image (jump edges). Although the edges can be computed from produced regions by contour following techniques, it was already pointed out, that a robust region–based segmentation must independently compute the edge information [25, 26]. If it is not the case, two distinct surfaces having the same combination of curvature signs may appear to be adjacent to each other. Without considering significant discontinuities in either depth or surface orientation these two surfaces could be merged. Usually surface curvatures are computed across discontinuities or are not calculated at all in the neighborhood of the discontinuities. In a similar way an independently computed vertex can increase the performance of edge detection. Without considering the vertex information two edges with similar features could be merged, although they belong to various surfaces.

4. *Merging*
   This is a region–growing, clustering or split–and–merge technique with surface fitting procedure or a boundary search with line fitting procedure. In case of a growing or clustering technique two regions or line segments are merged iteratively, the initial set of regions (line segments) being the set of pixels, until the approximation error becomes to large. In case of a split–and–merge technique the non–homogenenous image is splitted into four subimages until homogeneous regions are obtained only. It is tried to merge these regions by a growing technique.

5. *Low-level image description*
   The description process obtaines a characteristics of image primitves and it extracts relationships between primitives. At end each primitive category (junction, line segment, region) is characterized by classes (linear or curved line; planar, convex, concave region) and a each particular instance is distinguished by its attribute values. The attributes are of three types:

   - *geometric*: points, coefficients of curve– or surface–equations, symmetry axes, center of gravity
   - *topological*: connectivity genus, number of neighbors
   - *numerical*: elongation, length, perimeter, surface

   The relationships are either of topological nature (adjacency, boundary) or they represent occlusion relations.

B. Region detection approaches

Planar surfaces are detected [27–29] while exploring only the normal vector at each pixel. The unit vector normal is usually estimated to the tangent plane at a pixel. This plane is the best fitting plane (found by linear least square method) over an MxM neighborhood of the pixel. 3x3 or 5x5 neighborhhods were found good tradeoffs with respect to minimizing edge effect propagation and noise (quantization) effects. Also the position variable and planar fit error are computed.
Curved surfaces are detected while checking the normal vectors [30] or curvature measures [25, 31, 32]. In first case Hebert and Ponce [30] map the surface normals into the Gaussian sphere. Planar patches become clusters, cylinders – half circles od radius 1 and cones give smaller circles in the Gaussian space. The Hough transform is used to detect these shapes. The curvature measures of a surface are rates of change in surface normal as we move in a given direction from a point. Besl and Jain [31] pointed out that the signs of Gaussian– and mean–curvatures, K and H, are powerful enough to describe visible surfaces. While the above authors produce "fine" classifications of surfaces into several classes, Hoffman and Jain [31] claim, that the sensitivity of used measures to noise motivated them to concentrate on a "coarse" classification into planar, convex and concave surfaces only (Fig. 3.1).

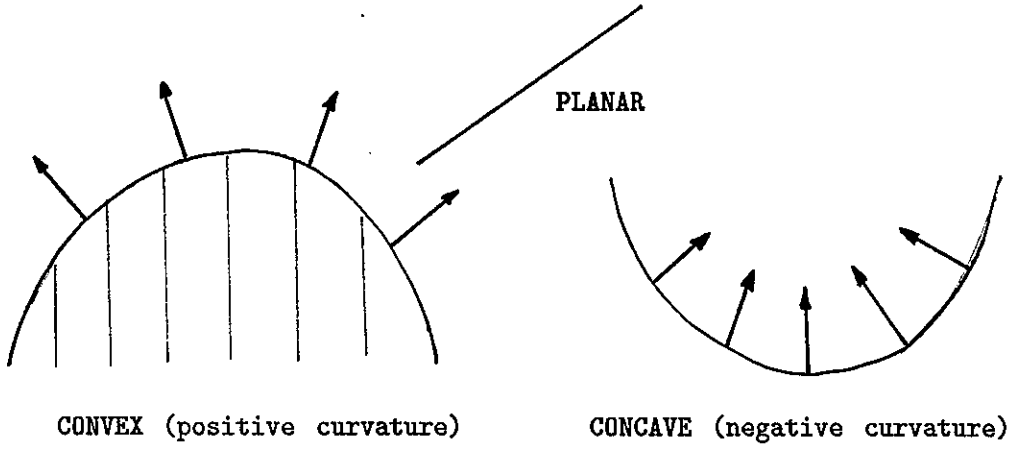


Figure 3.1: Main region classes in dense range images

C. Boundary detection approaches

Most commonly two types of edges are distinguished (Fig. 3.2): *jump* edges (called also step- or occluding-boundary) and *roof* (or crease edges or internal boundary).
Jump edges in range images are formed where the depth values are discontinuous. This case occurs,

if an object occludes another object or part of itself. Roof edges correspond to points over which surface normals are discontinuous. Convex– and concave–roof edges are distinguished. A third type of edge – the *smooth* (or virtual) edge can be also distinguished but basing on region features. Smooth edge consists of points where the surface normals are continuous but local curvatures are discontinuous.

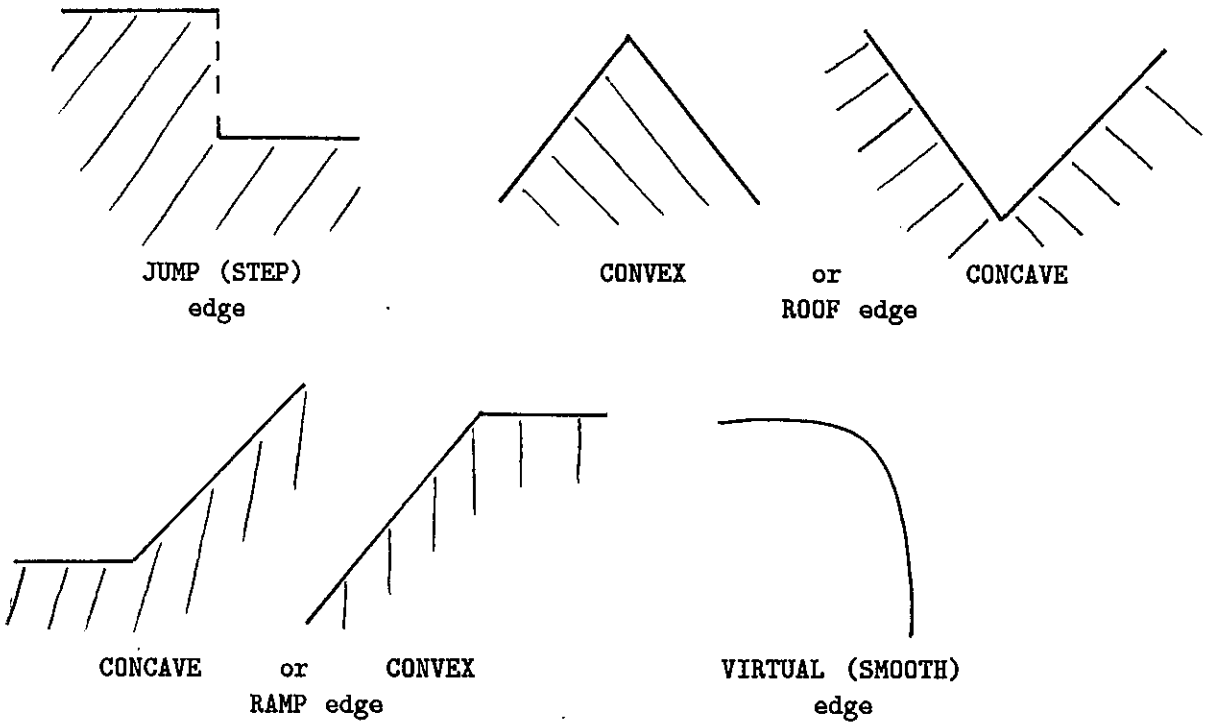


Figure 3.2: Edge types in dense range images

Before starting the edge detection [33–35] the intersections of potential step- and roof- edges are computed and labeled as *junctions*. Their features are: position in space and connectivity genus. Boundary points are merged iteratively (two neighboors) until the approximation error becomes to large or a junction point is reached. The boundary points are approximated by a set of linear or quadric elementary curves (occluding or internal). Their features are: equations of curves, length, elongations, symmetry axes.

3.1.3 Segmentation and description of sparse range data

Thus in dense range images all kinds of primitives can be computed independently in the segmentation step. In sparse (stereo) range data the surface information is not available. The range information is available only at the intensity discontinuities, typically at the object boundaries, surface discontinuites and surface markings. The data is not everywhere on the boundary available. Hence we must rely on both incomplete and imperfect 3–D boundary information only and 2–D regions. The surface information must then be hypothesized in the recognition phase.

Boundaries detected in sparse range data are classified into four classses [36] (Fig. 3.3):

1. *occluding* boundary – where the visible surface is only on one side of the boundary
2. *inner object* boundary – changes in surface orientation
3. *surface markings* – changes in surface reflectance

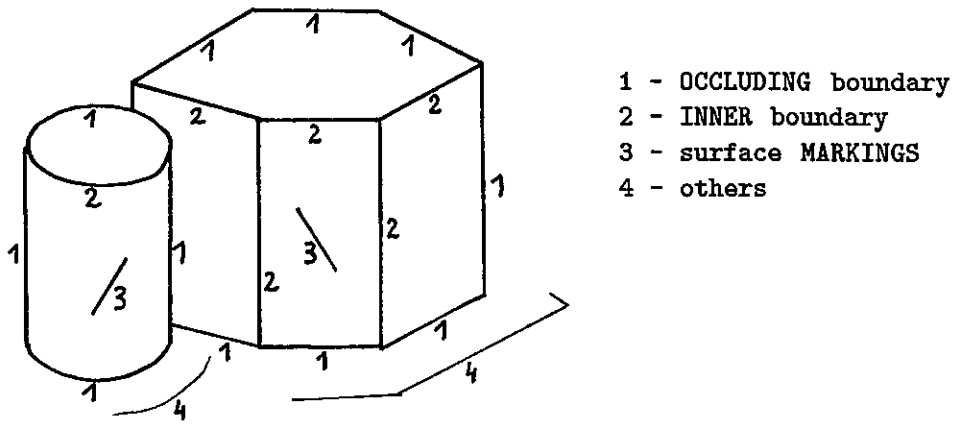


Figure 3.3: Edge types in sparse range data

4. others – due to noise, shadows, highlights etc.

Hence, among the segmentation data, there are boundary elements (from type 1,2) that are parts of visible objects, and boundary elements, which are superflous elements achieved by mistakes of the segmentation (type 3,4).

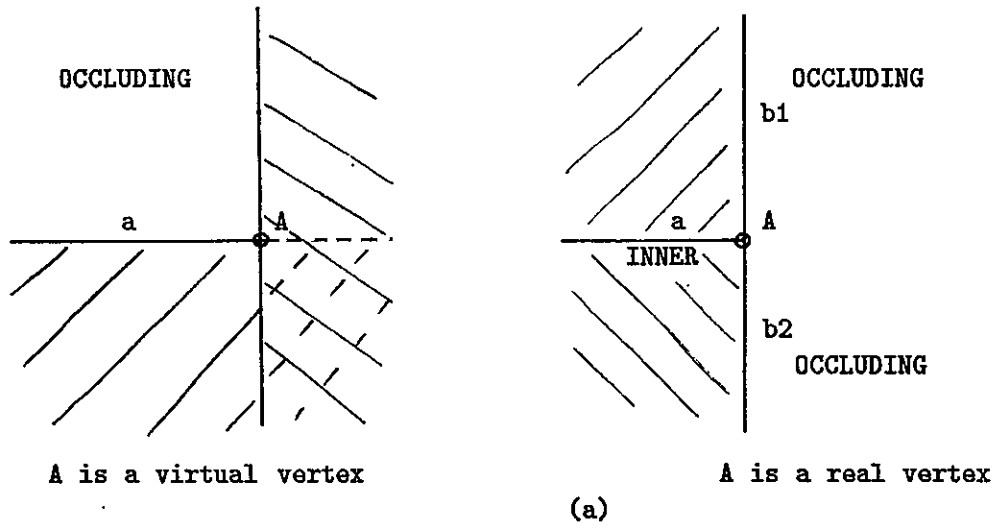
The output of segmentation for sparse range data is assumed to consist of tho types of primitives:

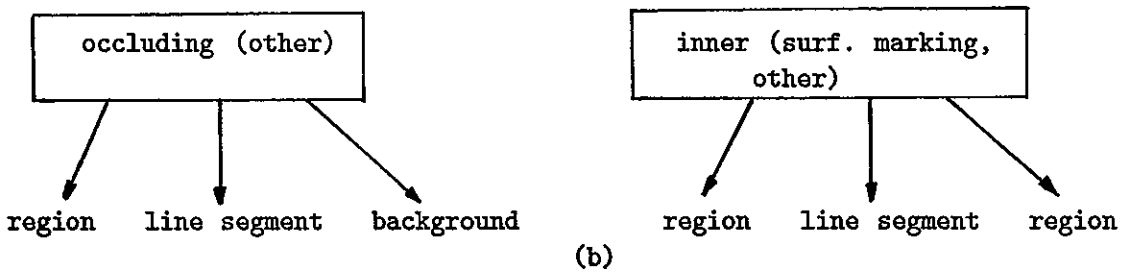
- 2–D and 3–D line segments - with positions of endpoints of each line in 2–D and 3-D spaces, and attributes such length, orientation
- 2-D regions - with attributes such as dimension, center point

The description process could be as follows:

1. find line relations - coplanarity, parallelism, convergence
2. find junctions - a junction is formed, when several segments meet at a point or when the distance between their extremities is within a preset threshold
3. find relations of the lines to regions
4. refine the line set - if a line segment delimits two various regions at each side, it should be divided in the junction into two collinear line segments.

On base of above relations the line segments can be classified into *occluding* and *inner*, and the junctions – into *real* or *virtual* (Fig. 3.4).





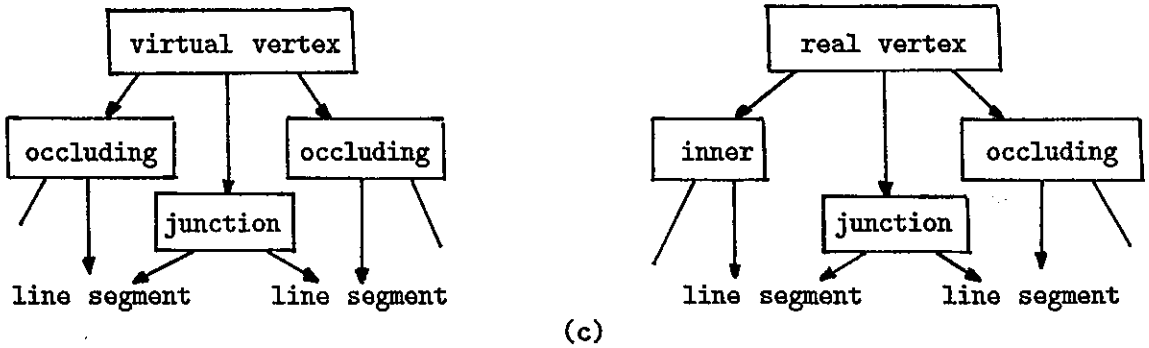


Figure 3.4: Secondary primitives generation: (a) example, (b) *occluding* or *inner* boundary, (c) *virtual* or *real* vertices

3.2 3-D object recognition

3.2.1 Prototype–free vs. model–based approach

There are two main approaches to 3–D object recognition:

1. the prototype–free partitioning of the scene into objects
2. the model–based recognition of objects

In the first approach we recognize objects without using explicit models of objects, i.e. we partition regions with surrounding lines and vertices into bodies and classify them. This is a data–driven analysis in which a domain–dependent knowledge is applied. In the second approach we try to match the image primitives in terms of model primitives and while basing on this matching to detect and to localize objects.

3.2.2 The sub–problems of object recognition

Goal of the object level–processing is to give a **scene description** of the image in terms of detected and located object instances and their relationships. In other words following questions should be answered:
– for statical analysis:
(i) WHAT objects are in the scene
(ii) WHERE are the objects located in space
(iii)HOW are the objects positioned against the viewer
– for varying–image analysis additionally :
(iv) HOW are the objects moving in the space

According to abstraction levels of the object model the one–frame analysis problem can be decomposed into sub–problems:

1. Classification of image primitives into model primitives
2. Visible Surface Reconstruction (ev. with texture analysis)
3. Single Object Recognition (in 2–D, 3–D)
4. Multi–Object Scene Description
5. Incremental scene description construction

In case of *range image (depth–map)* the recognition problem is already transformed into 3–D space. We must make a distinction between dense depth maps, obtained by active sensing techniques, and sparse range data, computed by passive techniques (i.e. stereo). In stereo depth information is available only at the intensity discontinuities, typically at the object boundaries, surface discontinuities and it also comes from surface markings. In dense range data only physical discontinuities and no surface markings are perceived. Hence from dense range data a description in terms of 3–D regions and line segments is usually obtained, whereas a sparse range data segmentation provides only 3–D line segments for significant points and 2–D regions only. The performance of the first two sub–problems may differ, in dependence from the type of available range data.

The method used for single object recognition is *recognition throu localization*, i.e. searching for a proper transformation matrix from range image points to model points.

The fourth task was rarely addressed so far. A search for consistent object configurations must be performed, which analyses the occlusion relations between objects and their parts.

Due to the complexity of both image and model domains one want to constrain the analysis space and to perform a selective and incrementally refined recognition – this is summarized by the last task.

3.2.3 Review of approaches

DATA & MODEL	
– Type of input data	sparse range image
	dense depth map
– Segmentation data	surface patches
	boundary edges and points
– Complexity of primitives	linear primitives
	curved primitives
– Object model	simple one-volume objects
	complex many-volume objects
– Scene domain	one–object scenes
	many–object scenes
– Time domain	one–frame analysis)
	varying–image analysis
RECOGNITION STRATEGY	
– Analysis processes	hypothesis generation, verification, prediction
– Matching type	model-to-image matching (model-driven)
	image-to-model matching (data-driven)
– Matching technique	relaxation
	Hough transform and clustering
	tree search
– Prunning tests	decoupled constraints
	coupled constraints
	transformation matrices
– Self–occlusion handling	null face–match
	partial–match
– Inter–object occlusions	like self–occlusion handling
	configuration tests (visibility computations)

Table 3.1: Some classification criteria for object recognition approaches

In case of sparse range data one usually tries to match the corner points or boundary lines detected in image with the model data [36](RAO& NEVATIA). If information about surface normals in selected 3–D points is available, these points can be matched with the model faces [37, 38] (GRIMSON& LOZANO–PEREZ). The creation of a dense surface by interpolation and the match with model faces is here a difficult task. Interpolation requires the knowledge of the object boundaries, that is the proper partitioning of image sections into objects. Moreover a surface description is not essential, if our ultimate objective is a volume–based object description.

A typical approach for dense range data is fitting surfaces to selected regions and next matching these surfaces with faces of the model objects [39–42] (BHANU, FAUGERAS& HEBERT, OSHIMA& SHIRAI, DHOME& KASVAND). The extraction of edges or corner points from the set of regions is also possible. Then these boundary elements can be matched with appropriate model elements also.

Both surface and edge data can be utilized in a composite matching with object models [43, 44] (BOLLES & HORAUD, UMEYAMA et al.).

An open question is the modelling of complex parts. Up to now only a standard face–edge–vertex description (one–volume objects) was used for object definition. Recently the CAD–models for multi–volume object definitions were proposed, but only the faces of such objects are computed and provided for matching with the image data [45]. An interesting approach to CSG–compatible model–based single object recognition was proposed recently by LIN & CHEN [46].

An other open question is the multi–object scene recognition, which must cope with unpredictable occlusion relations between objects. This problem was preliminary addressed for simple one–volume objects in [38, 43].

Some classification criteria for object recognition in range data are listed in Table 3.1.

Chapter 4

System outline

4.1 The assumed computer vision system

Two main analysis features in assumed computer vision system are:

- the *incremental analysis* mode
- the *heterarchical control* type

In the incremental mode the actual scene interpretation is refined and completed after new data is available. Thus in the *time domain T* the interpretation of some image at time t_k depends from previous interpretations for images at times $t_0, ..., t_{k-1}$. One static low–level image– or scene–description can be also incrementally constructed due to the selection of image sections (*image domain I*) or due to the selection of model elements (*model domain M*).

The incremental analysis in domain I means a selection of most important image sections or image features first and starting for them the statical analysis cycle. After an interpretation has been achieved, the selection of additional image sections or finer image features may be performed, depending from these analysis results.

The incremental analysis in domain M means a selection of the most characteristic objects first and starting the initial recognition cycle for them. The achieved scene description is next refined, while searching for additional objects or more specialized objects in the image.

The heterarchical type of control states, that the application order of particular analysis processes depends from the intermediate analysis results. There are two main directions of analysis – bottom–up (data–driven) or top–down (model–driven). A consistent image description and interpretation is made by a cooperation of processes of both types.

The processing structure of assumed analysis system is given on Fig. 2.1. Following processing stages can be distinguished:

Motion and Stereo – This is a multiple frame–analysis via motion and stereo algorithms in order to produce one depth map.

Scene cues generation – Some rules for the generation of scene cues are applied to the image in order to focus the high–level control on most probable scenes.

Prediction – This is a set of top–down model–based processes that propagate analysis constraints from current scene interpretation, for the analysis of next image. For example the *scene prediction* process selects constrained scene models on base of current scene cues and previous scene interpretation. The next object recognition stage is performed in a constrained M–domain induced by the predicted scene.

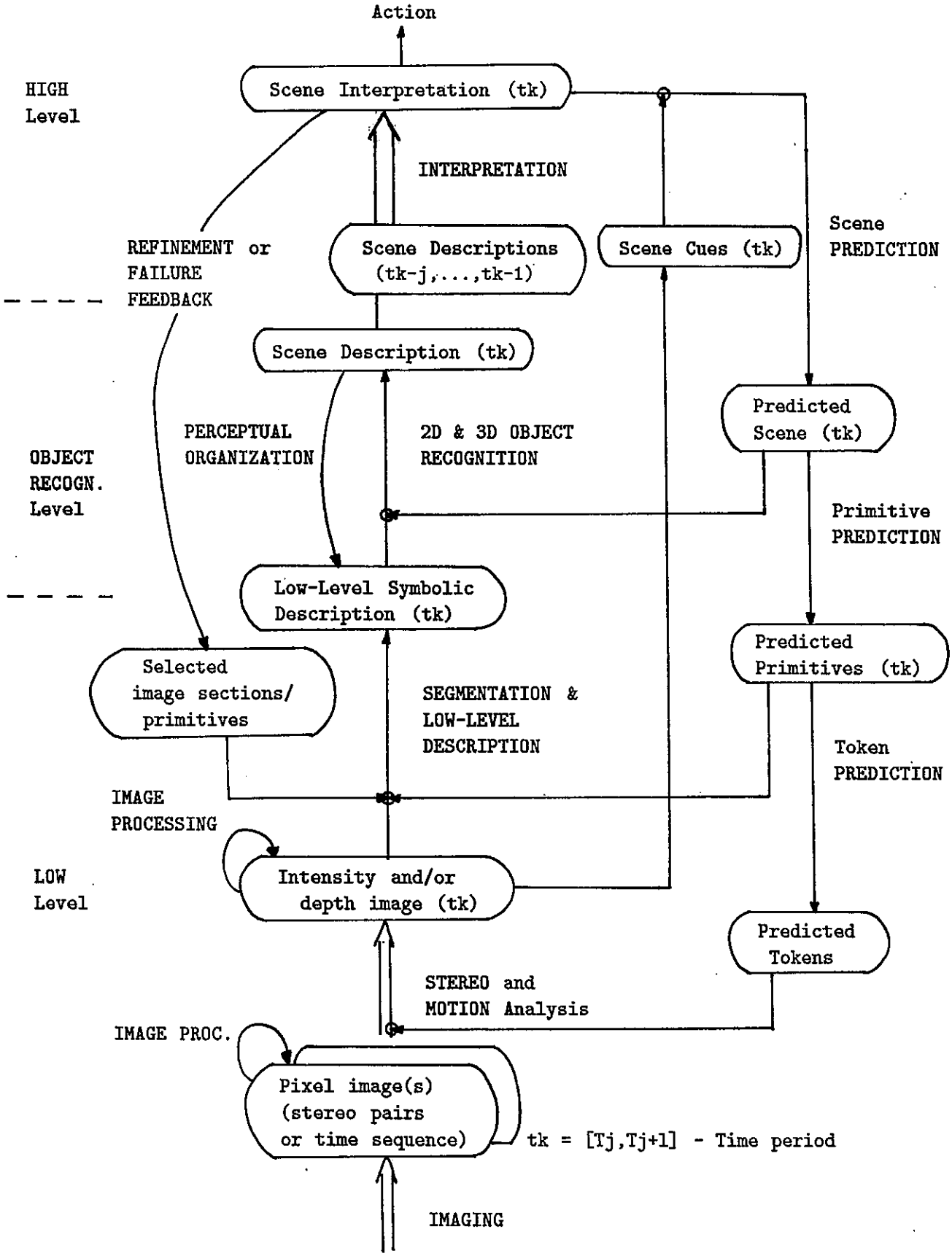


Figure 4.1: The processing structure of a computer vision system

Segmentation and low–level description – This process creates a symbolic low–level description by detecting the 2–D and 3–D primitives in the image. It can be performed in an iterative manner, in both I– and M–domains.

2–D and 3–D object recognition – It generates a scene description in terms of objects detected in the image.

Perceptual organization – It directs the merging of line and region primitives into larger aggregated structures, that are expected more closely to match the detected objects. This process is performed if the recognition succeds, but some primitves match the model only partialy.

Interpretation – This is a deduction process, which relates a human–like meaning to the current scene description and to the scene interpretations of previous images.

Refinement or failure feedback – If the recognition or interpretation processes end with failure or success, it causes the selection of new image features or additional image sections and a segmentation process for them is started.

4.2 Model–to–image matching in ERNEST

At the University of Erlangen–Nürnberg, a particular semantic net–based knowledge representation language ERNEST [8, 9] was developed.

4.2.1 Data structures

The network consists of three **node** types, five **links** and nine **substructures.**The three node types are: the "concept", the "modified concept" and the "instance". The data structures of the three node types are identical, except that a pointer to a function in a concept is replaced by the computed value in the corresponding instance. A modified concept is distinguished from the concept only by more restricted ranges for the attribute values.
Concepts, modified concepts and instances form different network partitions and they are associated by link types **instance** and **instance of.** Besides this relationship four other link–types define a nonreflexive partial order over the concept set: **specialization** (or inversive **generalization**), **part** (or **part of**), **concrete** , **model.**

A node is a complex data structure defined by a set of 26 **slots.** The most important and also the intuitively understandable slots are *attributes, parts (concretes)* and *relations.* Each one is itself described by a **substructure** called attribute–, link– and relation– description. The three next substructures refered by slots of the node structure are: *modality description, function description, identification.* Every substructure in turn is defined by a set of **items.** The remaining three substructures are refered by items of upper substructures: *adjacency, range, value.*

Besides the knowledge represented by node data structures and substructures, there are additional knowledge sources represented by **attribute–** and **concept– lists.** An attribute list contains the tuples – [concept, role, type], which are directly related to the model network. A concept list contains the names of all concepts having a set of tuples - [role, type].

4.2.2 Model expansion

Due to the relationship between the knowledge– and data– bases there is a *concept–based* instantiation or a *link–based* one. The first one means a generation of exact one instance for one concept, whereas the second one - exact one instance for one model link.

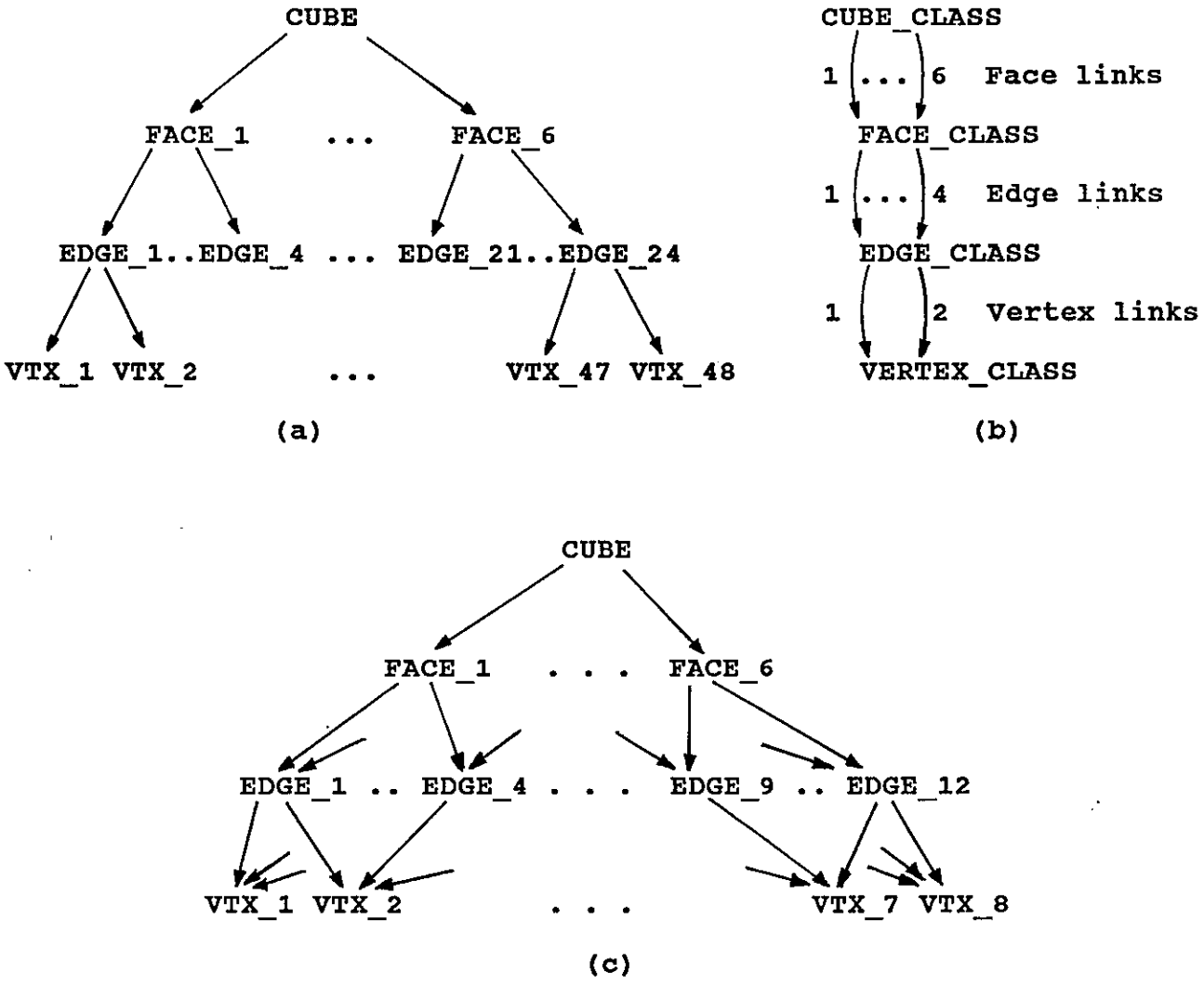


Figure 4.2: Example of a cube model: (a) a derivation tree, (b) a graph for a production set, (c) a mixed tree–graph model

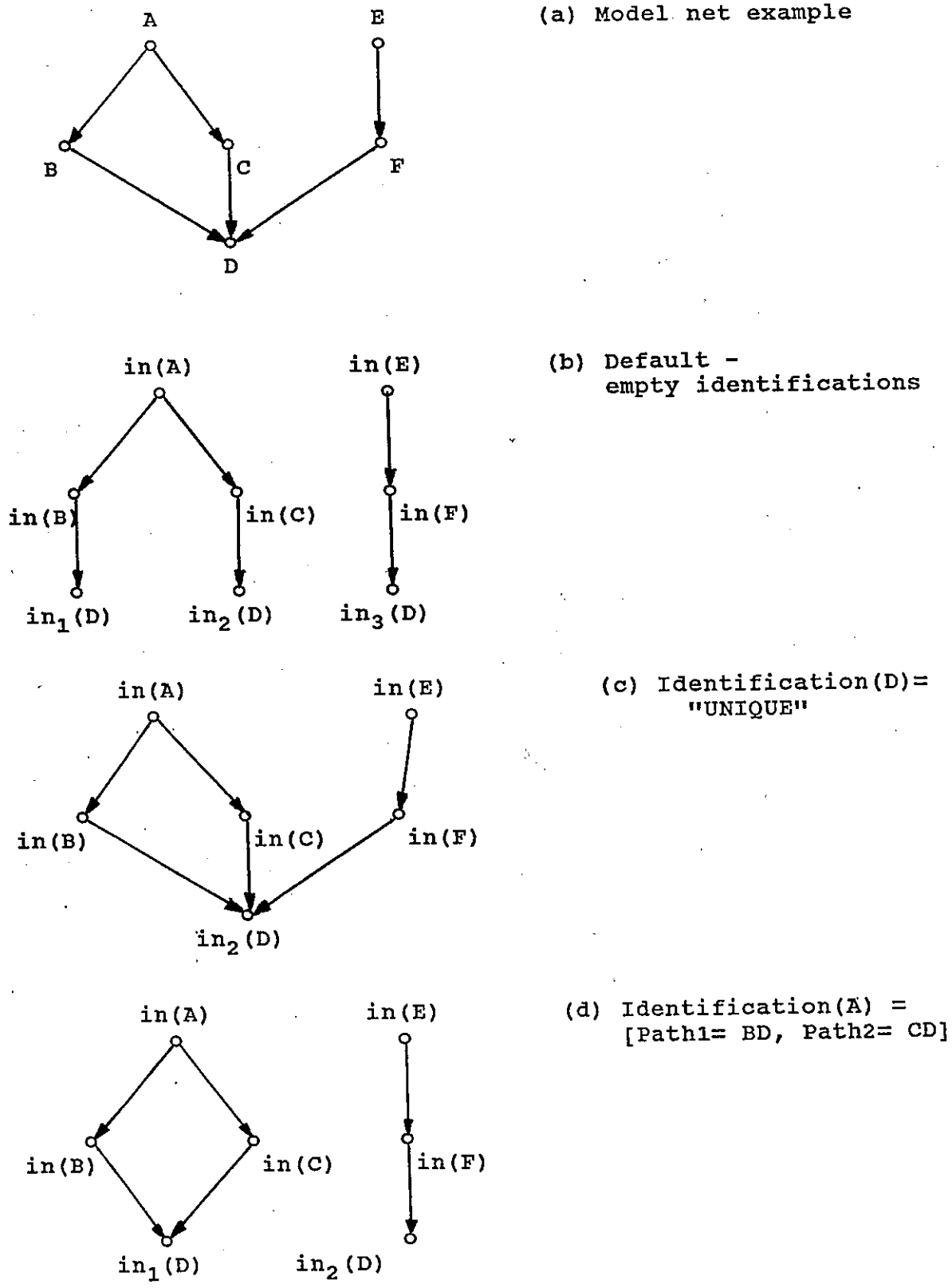


Figure 4.3: Concept–to–instance relationships: a) sample model net, b) link–based instantiation, c) concept–based instantiation, d) mixed instantiation

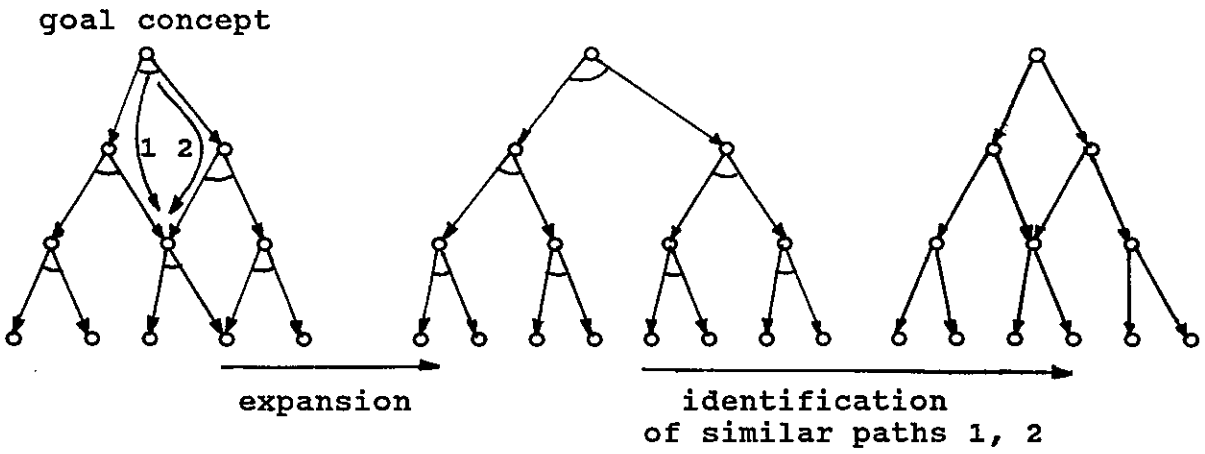


Figure 4.4: Expansion of a model concept

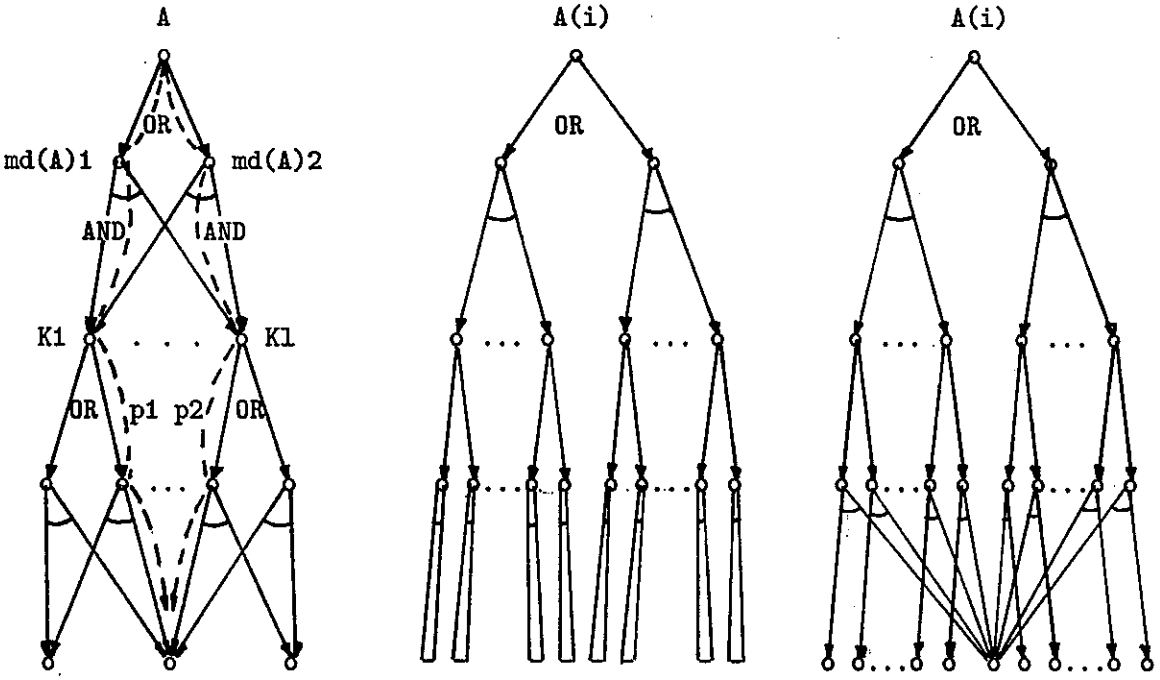
The difference between these models is similar to the distinction between the *derivation tree* and the *production set* of a generative (finite state) grammar. The ERNEST system supports both forms, but also a mixed third one (Fig. 4.2). This model concept–to–instance relationship is specified for each concept by its slot **identification** (Fig. 4.3).

In order to match the model with the image data, a subnet is top–down expanded, like for a link–based instantiation, but at the same time similar paths and unique concepts are identified and glued together (Fig. 4.4). Due to the *modalities* of concepts (denoted by md) the expanded model is an AND–OR graph and there are alternative solution graphs (*instantiation lists*) for the model–to–image match.

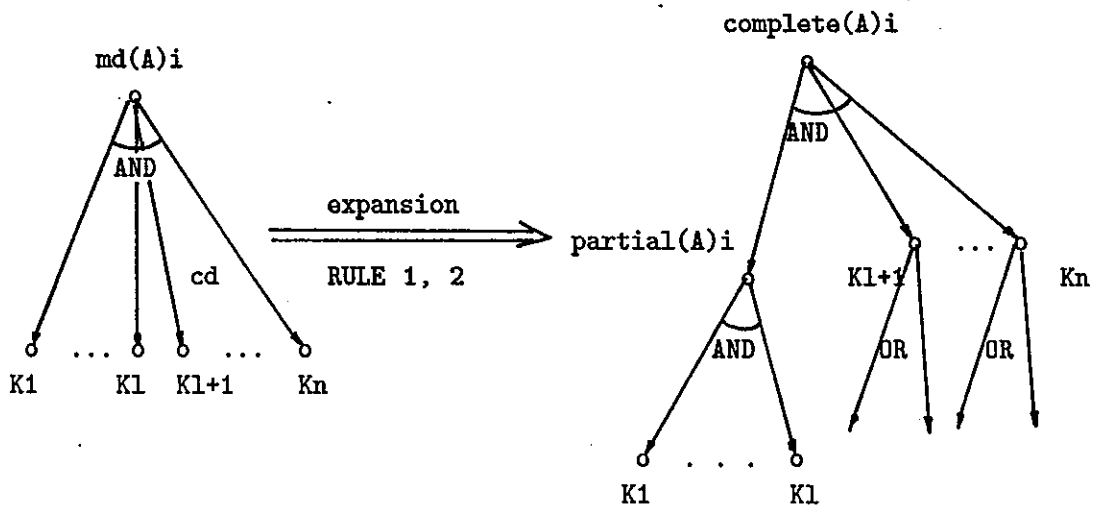
4.2.3 Inference rules

Computation of instances only depends on the syntax of the network (domain independence). There are six **inference rules**, that describe the use of knowledge. They are defined for the whole network, without respect to a task domain and are the basis for problem–independent control. The first three rules are for building up instances on base of instances and modified concepts. The last two rules are concept modification rules.

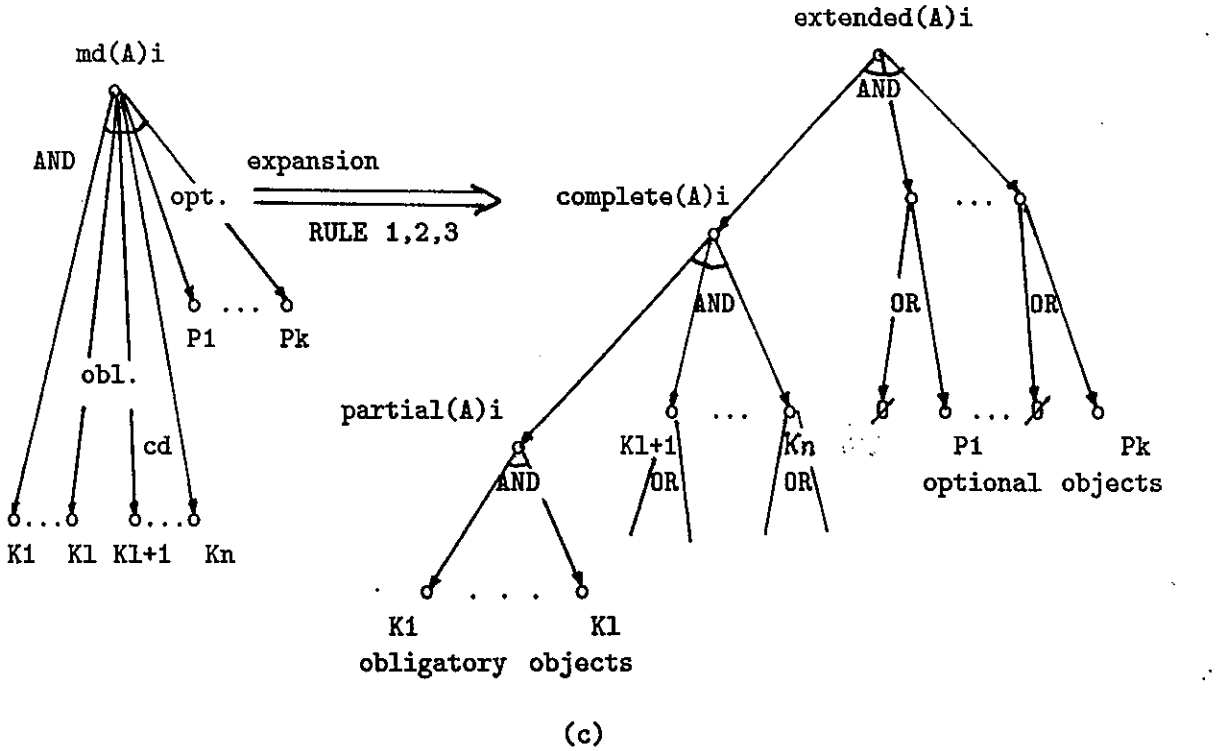
First a *partial* instance of concept A or modified concept mod(A) is computed by requiring only instances of the context independent parts and concretes (this is summarized by RULE 1) (Fig. 4.5(a)). Having the partial instance of A instances of context dependent parts M can be computed. Having instances of M the partial instance A may be *completed* (this is summarized by RULE 2) (Fig. 4.5(b)). RULE 3 checks whether there are instances of optional parts or concretes and generates *extended* instances of A from a complete instance of A (Fig. 4.5(c)). By the modification rules constraints can be propagated from bottom to up or from top to down in the knowledge hierarchy. If an instance of some part (or concrete) of concept A is available, a more refined range of attribute values of concept A may be computed (RULE 4). On the other hand, having a modified concept of A, this may in turn be used to restrict attribute values of its parts or concretes (RULE 5).


A
A(i)
A(i)
OR
md(A)1
md(A)2
AND
AND
K1
Kl
OR
p1
p2
OR
OR
OR
expansion
RULE 1
identification
of similar paths
AND/OR graph
Model
AND/OR tree
AND/OR graph
Expanded Model

(a)


md(A)i
AND
cd
K1
Kl
Kl+1
Kn
expansion
RULE 1, 2
complete(A)i
AND
partial(A)i
Kl+1
Kn
AND
OR
OR
K1
Kl

(b)



(c)

Figure 4.5: Concept expansion by: (a) RULE 1, (b) RULE 1 and 2, (c) RULE 1, 2 and 3

In order to incorporate results of initial segmentation in a bottom–up manner, RULE 6 was introduced. If from initial segmentation a primitive having an attribute with a certain and type is obtained, this may indicate the occurence of the concept in the corresponding row of the attribute list.

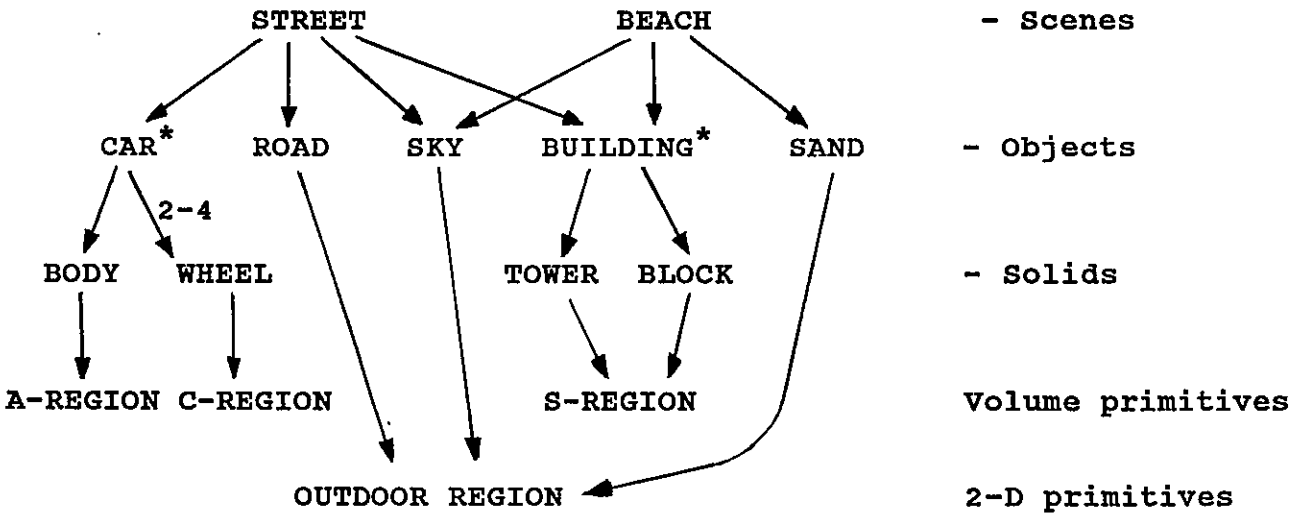
The above six rules in connection with the A^*–tree search algorithm form the skeleton for different control strategies [47–49].

4.3 Specific requirements for multi–object analysis

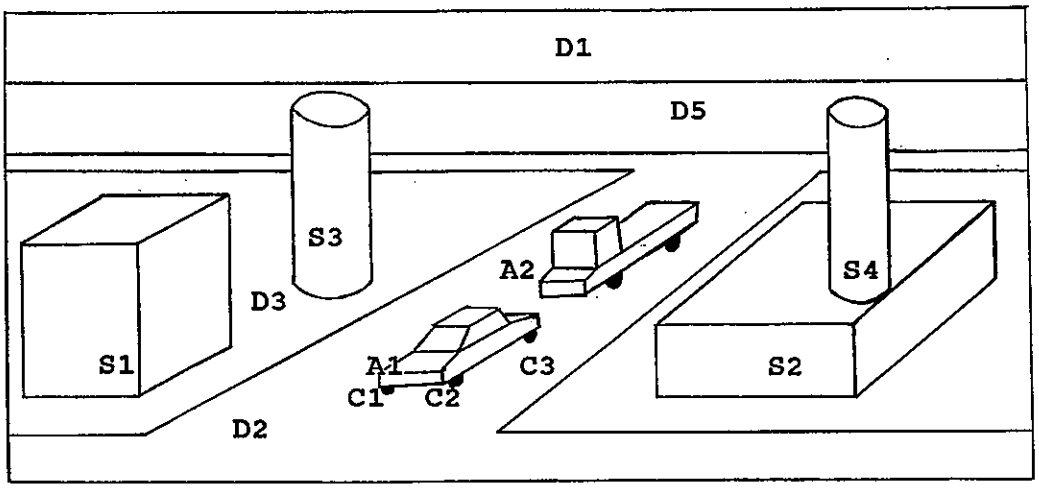
4.3.1 Local search spaces for unlimited/iterative links

A potentially *unlimited* number of objects can occur in a scene. Both the number of object instances and their mutual geometrical relationships can not be pre–determined. For example in general it is not possible to specify the exact number of cars in a "street scene" and to predict the movement of cars. At other side *iteratively* constructed solids (from volumes) or faces (from regions) can exist. Contrary to unlimited parts two iterative instances are usually related each other by some geometrical relationship.

It is proposed here to mark these unlimited/iterative links in order to iterate the instantiation of concepts refered by them. Originally the search space was expanded if the application of instantiation rules resulted in more than one created instance of a concept (or modified concept). Now for each instance I(C^*) of an iterative/unlimited object C^* generated in current search node N two successor nodes in the search space are created – one node O from which the next concepts will be instantiated (as usual) and one node O^1 from which the instantiation of last concept will be repeated. Image primitives available at each node in the search space are provided by a list PRIM. If a node from type O^1 is created, its set PRIM(O^1) holds only these segmentation results, which are not interpreted by instances from INST(N). See the example on Fig. 4.6 – for simplicity a low–level description over volumes is assumed.

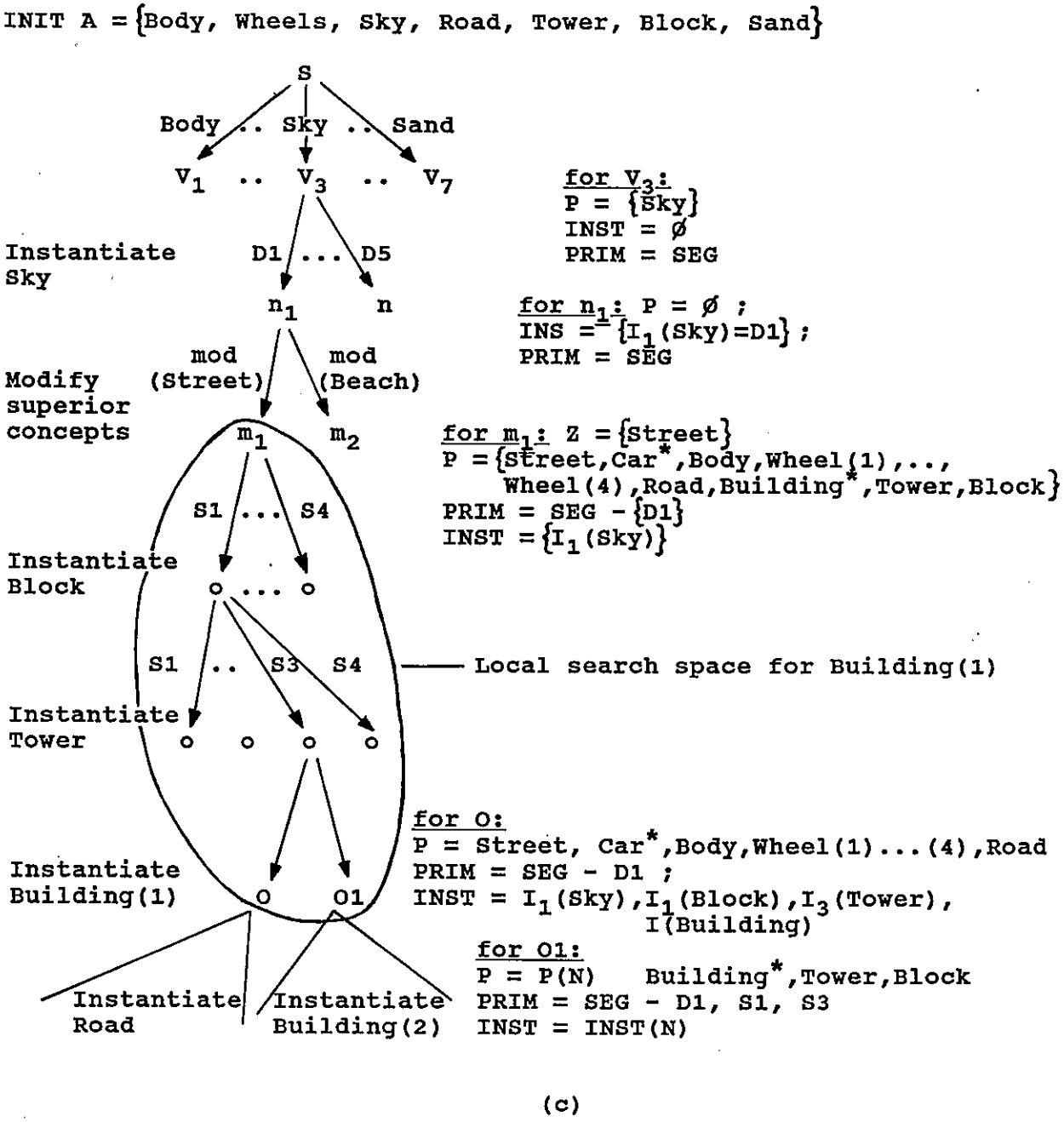


(a)



SEG = {D1,...,D5, S1,...,S4, A1,A2, C1,...,C6}

(b)



(c)

Figure 4.6: Iteration of search: (a) a simple model, (b) scene, (c) search tree

As there are obligatory and optional links available, we will restrict the unlimited/iterative links to be optional only.

4.3.2 R–concepts for global verification

In scene analysis we must cope with incomplete image data and a hypothesis–generation–process should be added, which explains the occlusions of objects. The verification means testing the visibility relationships between hypotheses. In ERNEST relationships can be represented by a relation description of a concept or by concepts themself. But our objective is to check the mutual consistency of all instances of some concept(s), without respect to the objects from the expanded model, which they are instances of. For example multiple instances of one concept can exist, due to unlimited/iterative links (Fig. 4.7(a) or due to various superior objects from the expanded model (Fig. 4.7(b).

In order to represent generic relations between instances (such as viewpoint dependent relations) so called *R–concepts* are introduced. They are represented by the same data structure as usual. The difference is in their use only – they are activated by RULE 1A. The features of R-concepts are:
1) they do not control the creation of instances of its part concepts, i.e. during model expansion there are not made distinct objects for parts of an R–concept
2) instances of a R–concept are non–competitive, i.e.the search space is not extended, while creating multiple instances of one R-concept
3) the number of R–instances is data–dependent, i.e. it is not searched for one instance of a concept (or link to a concept) from the model, but for all possible instances
4) the identification slot, not useful during expansion, restricts the set of instances for which the instance of an R–concept is to be created

The new RULE 1A is as follows:

FOR(each instance tuple J in INST consisting of instances of concepts from the premise of A)
IF (no other instance tuple J' equal or symmetrical to J was already applied for the creation of an instance of A in INST)
THEN apply RULE 1 for J

4.4 The vision model

Standard object representation schemes have been mainly designed for constructing (CSG – constructive solid geometry) and display (Boundary representation). They were only seldom applied in vision systems. If already used, then only for a deductive generation of vision representations and this transformation was not invertible. Hence it would be important to use the same representation scheme for construction as for vision, or a similar with direct correspondence. But even when such an object construction scheme appropriate for the vision system is established, it must be completed by additional knowledge. A vision–oriented model is a redundant one – each description entity is explicitly listed under several classifications and various partial representations of it must be considered (due to occlusion, missing items, finite viewing accuracy).
Three representations of man–made objects are distinguished:

A) Standard *CSG–representation* in a CAD system

B) The *model–scheme* in a vision system – a mapping of description entities from the standard CSG–representation into vision description entities

C) The *object model* in a vision system – knowledge for the recognition of description entities

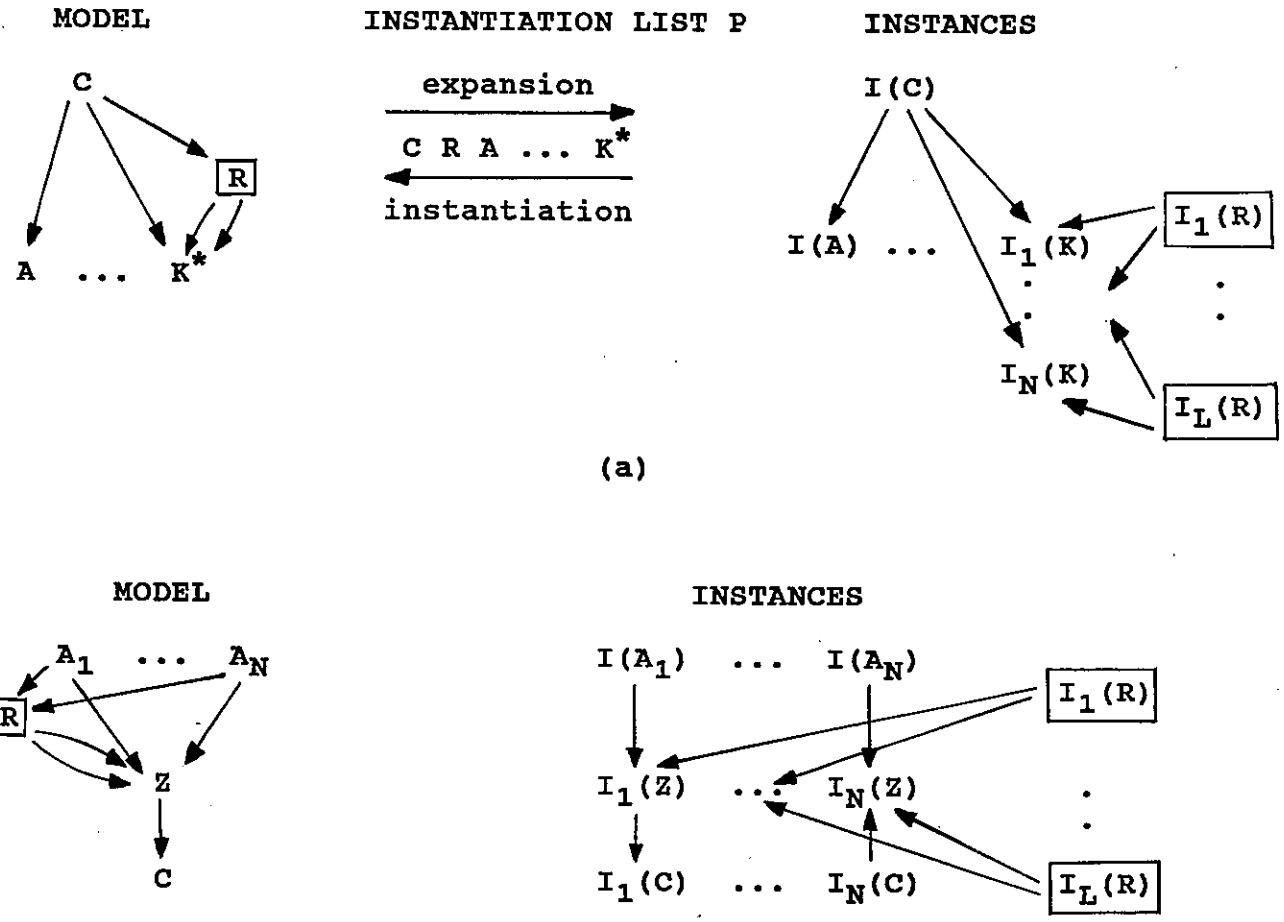


Figure 4.7: Instantiation of R–concepts

The objects from the *vision model* (schemas B) and C)) are defined in terms of solids, which are directly related to the CSG scheme (A).

The vision model is often aided by so called *attribute tables* (or key features), which allow the prediction (cueing) of the description entities directly from the image primitives.

4.5 The control strategy

We use a semantic net–based system for image understanding, where the knowledge is activated by several rules. The recognition of multi–object scenes is modelled as a onptimal forward search in the space of competitive partial scene descriptions. During the search following recognition processes are performed:

- bottom–up *cue generation*
- top–down *prediction*
- bottom–up *analysis*
- top–down *synthesis*

Let us notice, that the directions refer to the model hierarchy. Cue generation means the filling of attribute tables on various abstraction levels directly from the segmentation data. Prediction is the top–down creation of the instantiation list for a goal concept, combined with the creation of modified concepts in this list. The analysis and synthesis processes are expressed by the application of instantiation and modification rules to the concepts from the expanded sub–model.

The overall state space for the recognition problem is the result of a *data–to–model* match and is here called the *data–dependent* space. A control strategy is ERNEST, which tries to find the best terminal node in this space, is described in terms of a forward search in a distinct state space, which we call the *model–dependent* space. Both spaces are *implicit* given only.
The expansion of nodes in the model–dependent space is controlled by the A^*–tree search algorithm. The base of our control startegy is the *bi–directional* control [9] (see Table 4.1), which performs an *incremental model–to–image* match. In this context the term "bi–directional" does not correspond to its classical meaning of a bi–directed search (combined forward and backward search), but it refers to the computation flow as mapped onto the hierarchical model. There is a bottom–up (or *data–driven*) selection of goal concepts and a top–down (or *model–driven*) instantiation of a selected goal. Strictly speaking the instantiation process is simplified into a two–step approach: first a top–down model expansion of the goal by inversive application of RULE 1 and RULE 2, and next the bottom–up creation of instances for the fully expanded goal.

initialize the search space S by the root node R from segmentation results and concept tables compute a set A of concepts corresponding to segmentation results		
FOR each concept K in set A DO:		
	generate one successor node V of the root R in the search space S apply RULE 6 to K to compute a modified concept mod(K) the judgement of V is equal to the judgement of mod(K) generate list P(V) for concept K and node V Z(V) = K	
WHILE specified degrees of concretes and parts have not yet been achieved DO:		
	select the node N with best judgement in search space S	
	IF	P(N) is not empty
	THEN	choose the last concept C on P(N)
		instantiate C by using RULE1 and RULE2
		FOR each instance of C DO:
		generate one successor node O of N compute judgement of O by rules of algorithm A^* generate list P(O) for the node O remove C from P(N); Z(O) = Z(N)
	ELSE	determine a set B of superior concepts of Z(N)
		FOR each concept K in set B DO:
		generate one successor node V of N in space S apply RULE4 to compute a modified concept mod(K) the judgement of V is equal to the judgement of mod(K) generate list P(V) for K and V; Z(V)=K

Table 4.1: The basic bi–directional control [9]

Chapter 5

The model

5.1 The model–scheme

A 3–D object is generally described by means of topological and geometrical entities. The topological entitites stand for sets of points in 3-D space and give the roles they take in the construction of an object. If we want to order these points into classes, then geometrical entities are used (line-, surface-, volume- equations). According to the class of equation which constrains a set of points related to a topological entity, we have various **description elements.** These elements are represented by nodes of a semantic network. Thus the topology induces a *vertical* hierarchy and the geometry – a *horizontal* (or specialization) hierarchy of the model scheme (Fig. 5.1, Tab. 5.1).

Topological hierarchy:

- *Level 7*
  The highest topology level contains *scene* concepts. A particular scene consists of a set of *3–D objects* and of a set of *outdoor background* elements (e.g. sky, see, forest etc.). The parts of an 3-D object can be natural or man–made.

- *Level 6*
  Each 3-D object is defined in terms of a set of *solids*. The solid parts of an object can dynamically change their relative positions (i.e. rotate one around other). Only these similar solid parts can be symmetric, which are at each time symmetrically located around the object center point.

- *Level 5*
  For man–made solids the CSG–representation scheme is used. There can be the inside or outside of a solid visible. By inside views one handle the *indoor background* of a scene. A solid is defined by a CSG-tree. The leafs are *shell* (volume) parts, the non-leaf nodes are Boolean operations of union, difference and intersection and the arcs represent space transformations of the operands to the ancestor. In opposite to the object–from–solid construction, shells of an solid are in stable position and their structures are also determined. Only relative dimensions of parts can vary from one shell instance to another one. Each solid has its own coordinate center. Some shells can be symmetrically positioned around this center and if they are of the same type and dimensions they may be symmetric.

- *Level 4*
  Shells (volumes) are sets of connected points in space bounded by *faces*. Faces can be either directly specified by a boundary representation or computed from a sweep representation of the shell.

- *Level 3*
  A face consists from an outer *loop* and a *textured surface patch*. The textured surface patch

Level	Concept class	Links	Attributes
1	intensity region		
1,1+	3-D line segment		two end points, curvature param., type
1	3-D region		dimension param., center point, curvature measures
1,1+	vertex		point
2	edge	line segment	line equation param., two corner points
3	surface patch	region	surface parameters, fit error
3+	texture	surface patch 2-D region	symmetries of texture
3	loop	1 to n edges	corner points, loop/hole identifier
3+	text. surface patch	surface patch texture	hole number
4	face	1 loop 1 to n holes text. surf. patch	surface equation parameters
4	background	loop region	
5	shell	faces	model-to-image space transf.(3xRTS), iter. instance numbers, dim. of iterative links max. and min. bounding boxes
6	solid	shells background regions	origin, model–to–image space transf. (3xRT)
7	object	solids	model–to–image space transf. (1xS, 3xRT), max. max. and min. bounding boxes solid movement vectors

Table 5.1: Topology levels of the model

constrains the symmetry class that the face derives from its loop. Large surface patches, which cross the Window are supposed to be parts of indoor background.

- *Level 2*
  Surface patches are the result of classifying the *regions* given in image into several types according to the curvature measures. A surface equation is fitted to each surface patch with the fit error. Loops are closed curves, which consist from *edges* satisfying a minimal invariant: if $E_1, ..., E_n$ are edge-parts of a loop, then 1) the starting vertex of E_{j+1} and the ending vertex of $E_j (j = 1, ..., n-1)$ are the same, and 2) the starting vertex of E_1 and the ending vertex of E_n are the same. One specific loop – the WINDOW – represents the viewing border and is computed from the camera parameters. The edge parts of a loop can be symmetric giving equivalent loop instances (i.e. a rectangle has two symmetric instances in each equivalence class - the second one is derived from the first one by rotation by 180° around loop's normal vector). If an intensity image is provided a texture analysis of detected surface patches results in a textured patch. This task is better constrained than in 2-D case because not only intensities but also pixel normals are available. The texture can have symmetries, which constrain the symmetry set of a patch.

- *Level 1*
  An edge consists of a *line segment* and two *vertices* or three points. An opened edge is identified with the portion from the start to the end vertex. If the curve is closed an additional "near" point determines the direction. Vertex parts of an edge are symmetric.

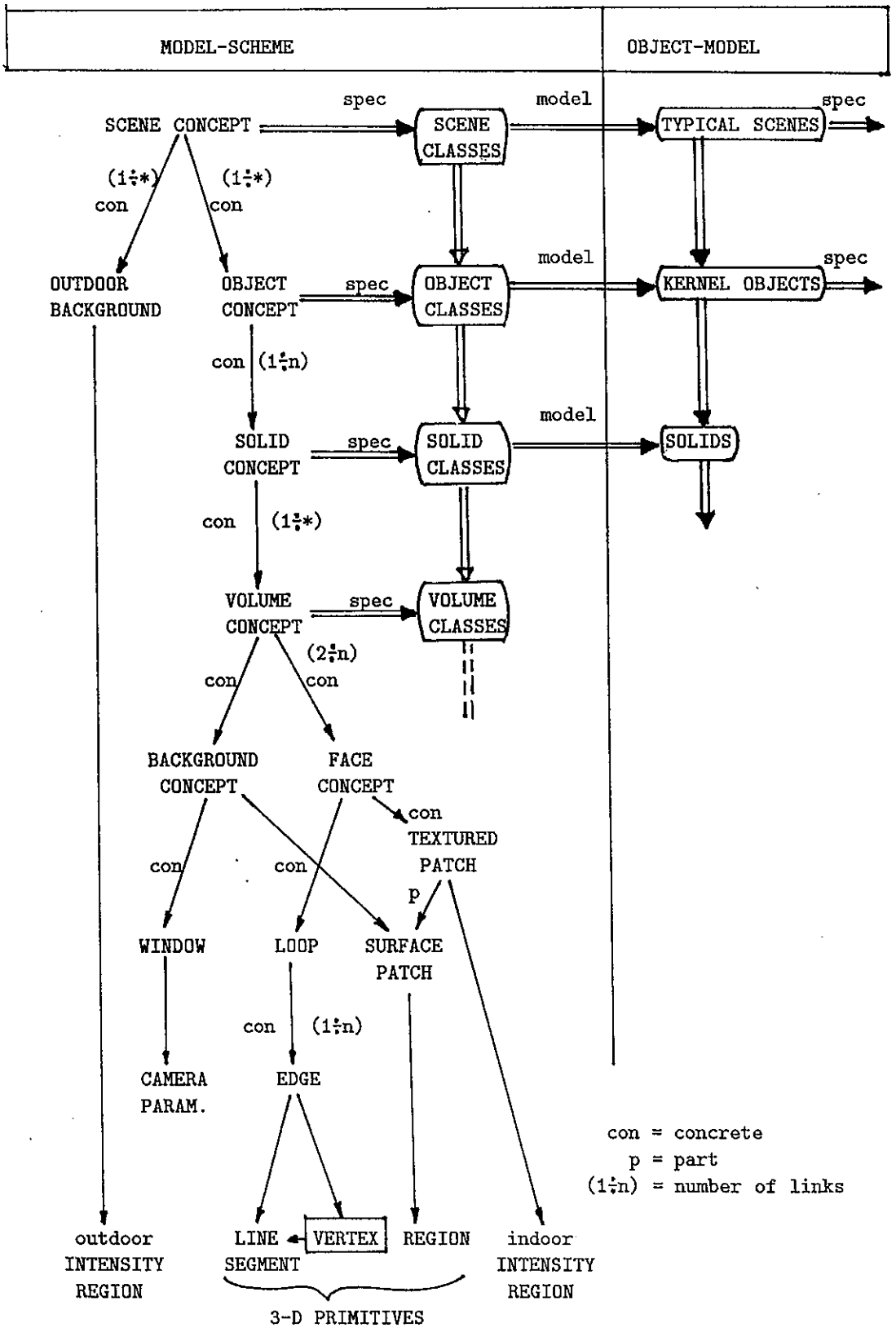


Figure 5.1: The Model-scheme overview

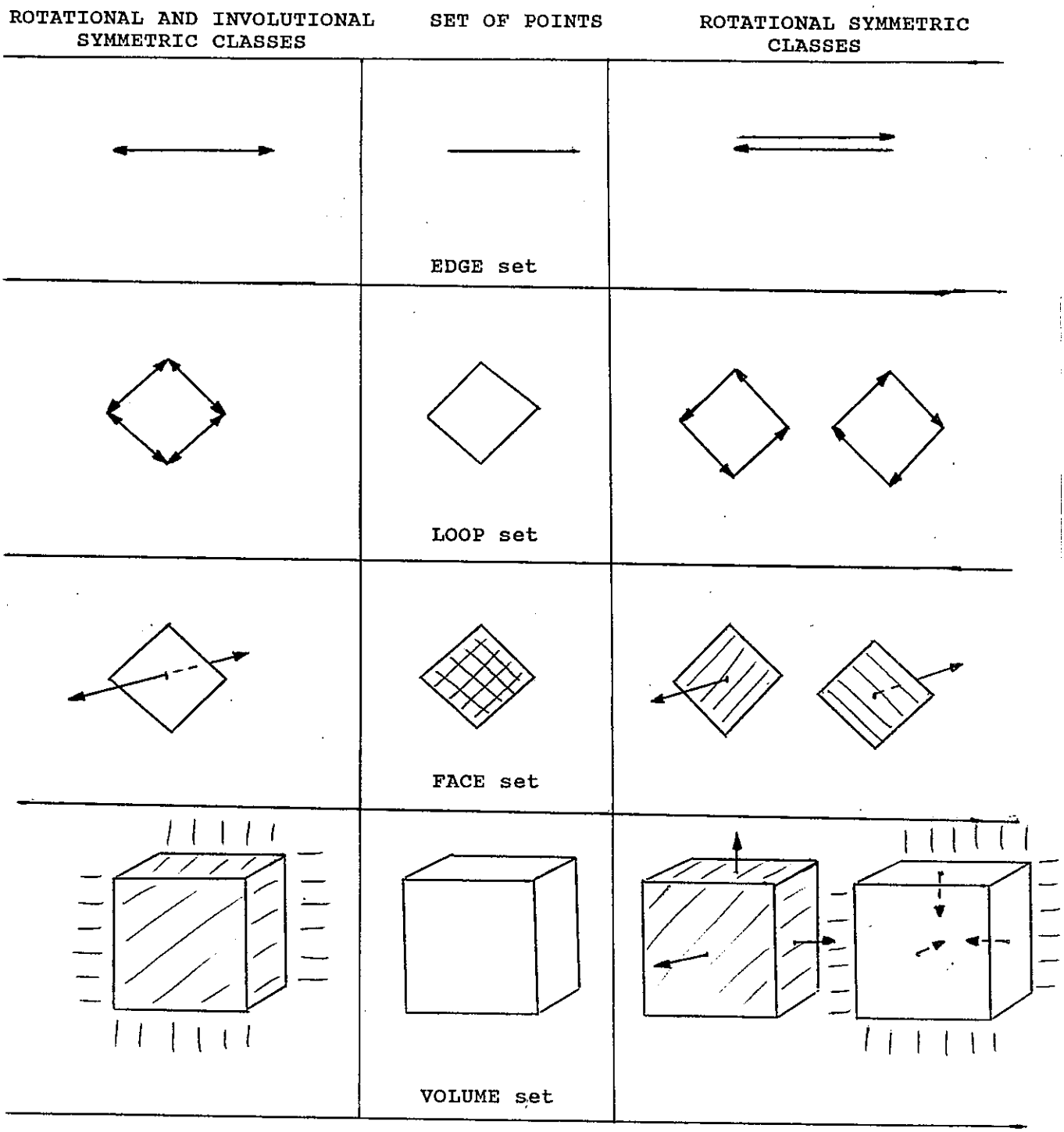


Figure 5.2: Mapping description elements to the model–scheme

- *Level 0*
  The lowest level contains 3–D primitives like line segments and regions, as well as 2-D intensity regions (if an intensity image is available). Sometimes connected groups of regions (blobs) are considered as primitives. At one side the primitives have numeric attributes and at other side they are related to range pixel sets – line segment is is a chain of pixels and region is a set of connected pixels.

Symmetries of description entities

A description element is *symmetric* if its shape is unchanged under an affine transform. Transforms of interest fall into two categories – *rotational* transforms and *involutional* transforms. Only first category symmetries preserve both shape and topology features. The involutional symmetries preserve only shape. We consider two mapping types between description elements and model–scheme concepts (Fig. 5.2):
(1) One concept represents both rotational and involutional symmetry classes of a description element
(2) There are distinct concepts for involutional symmetriy classes of a description entity, if they are not at the same rotational symmetry classes.
A third type of mapping, in which for all elements from a symmetry class distinct concepts are provided, is computationally expensive.
The second type of mapping is chosen here. Thus an edge concept represents two opposite directed line segments and can be shared by two faces of an solid as well as two faces of an involutionally symmetric solid. Two possible geometrical equations are implicite related to an edge concept, which represent directed edges. A loop concept is already part of one face. A distinct concept is used for an involutionally symmetric loop. The concepts for faces and cubes represent rotational symmetry classes only.

5.2 Mapping the CSG–tree to the vision model

5.2.1 Representing the CSG–tree in the model scheme

A single solid concept corresponds to one class of the CSG–trees. The shells are *concretes* of solid concepts and they correspond to leafs of the CSG–trees (Fig. 5.3). The operations from the CSG-tree are represented in the *analysis relation* of the solid concept and the transformations related to arcs of the CSG–tree are specified in the *link descriptions* of the concept.

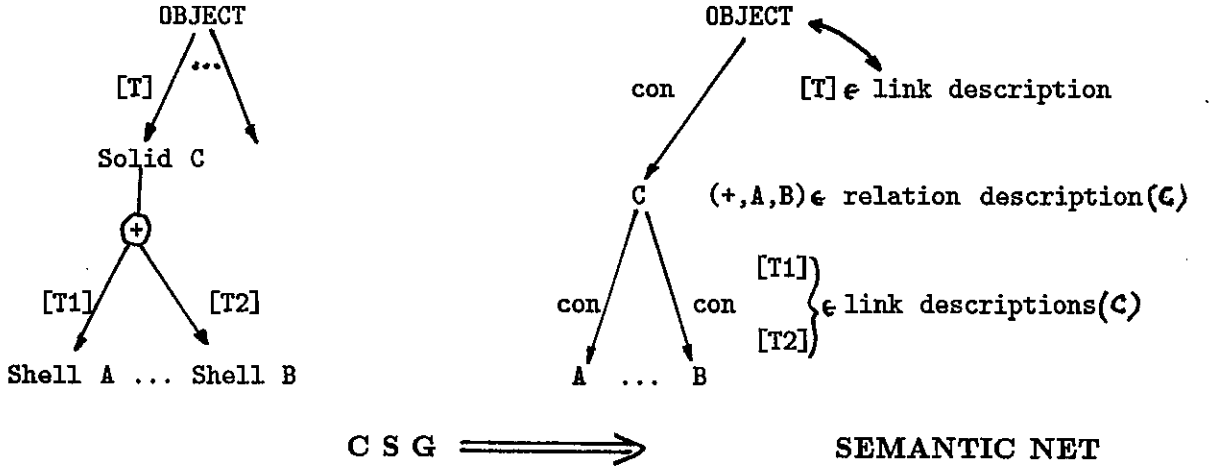


Figure 5.3: The solid concept for a CSG–tree

A solid class definition with a restricted iteration of union operations is allowed (Fig. 5.4). Each iterative link is labeled by "*".

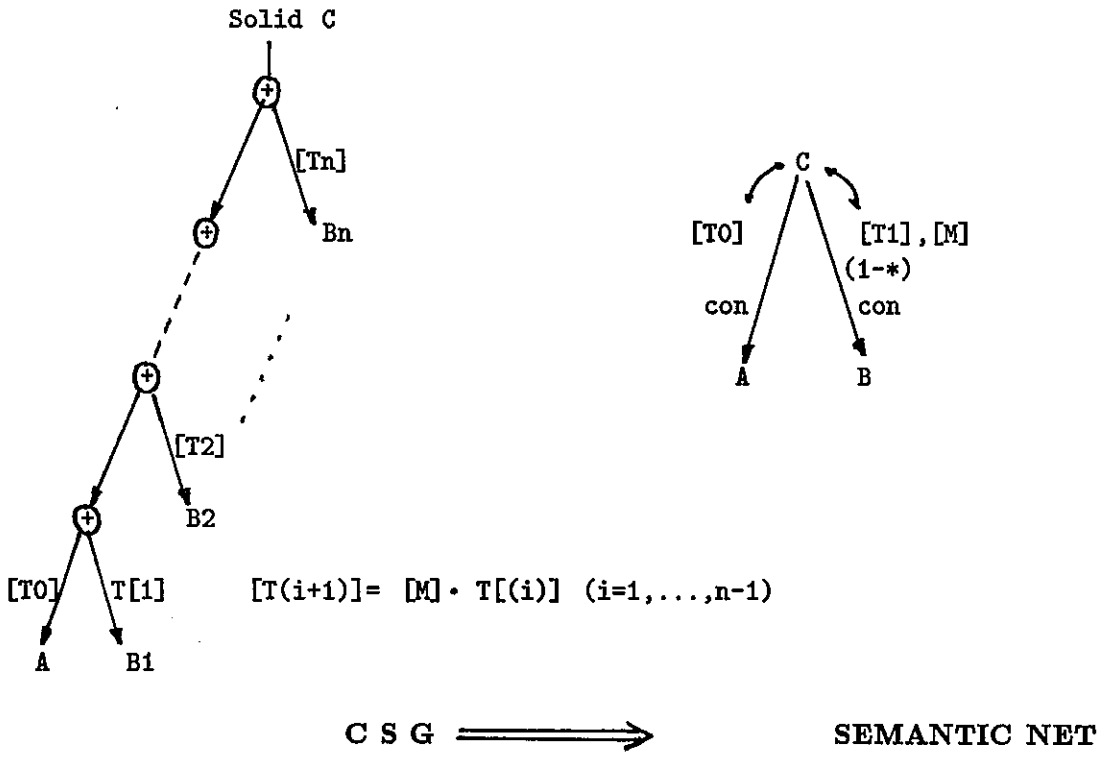


Figure 5.4: An iterative concept for a class of CSG-trees

The volumes are either *positive* or *negative*. If in a top–down traversion of the CSG–tree it appears, that the leaf volume is a subtractor at most once or an odd number of times, then a negative volume is linked to the solid concept. If a leaf volume is not a subtractor at all or it plays this role an even number of times, then a positive volume is provided in the model. In the example shown on Fig. 5.5 the volume D is twice a subtractor. Hence it is searched for instances of a positive volume D in the image. The volume C is a subtractor one time, hence a concept "negative C" is provided in the model. The transformations represented by arcs of the CSG–tree are bottom–up multiplied in order to derive a composite transformation of the coordinate system of each volume concrete to the solid system.

5.2.2 Mapping the boundary of a solid class to the object model

In order to represent the solid class-dependent face distortion, the boundaries of each volume of the solid class must be pre-computed. For this goal the CSG-tree is traversed bottom-up and the operations are solved. The solutions of Boolean volume operations are expressed in terms of bounding faces [17] (Tab. 5.2). Two predicates – *outer* or *inner* – are specified for each face of A and B due to the binary relation between them. A face from A which is outer of the volume B is taken with positive sign into the union and difference volumes, and is removed from the intersection volume. In turn a face from A which is inner of the volume B (or on the border) is taken with positive sign to the intersection volume, and is removed from the results of other operations.
Hence the object model provides (contex dependent) face concretes of a solid class for each positive volume concrete (Fig. 5.6). These faces are characterized by tree face predicates:

A_{face} – a full face (non-distorted)
B_{face} – a completely removed face
C_{face} – a distorted face

In case C the face concept has two concrete links – to the loop and surface patch concepts. The loop

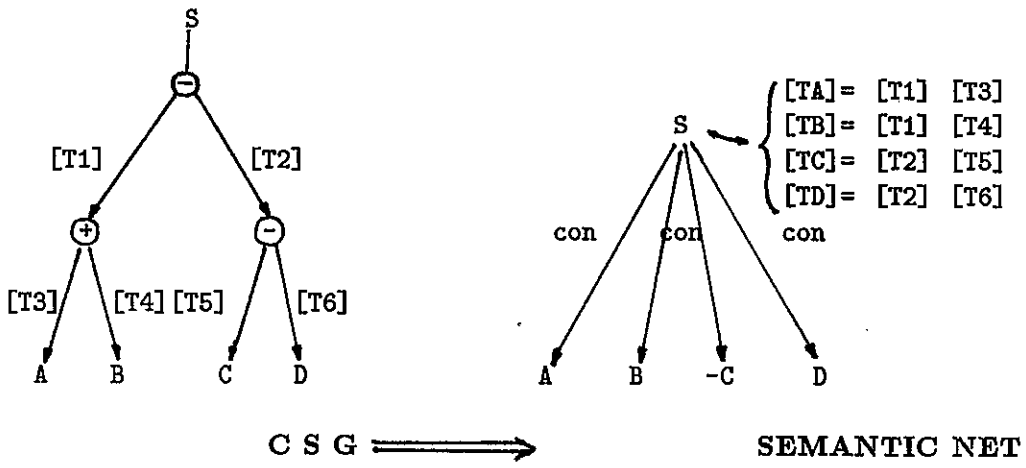


Figure 5.5: Example of a negative volume–link

Operation	A		B	
	outer	inner	outer	inner
$A \cup B$	+	not	+	not
$A - B$	+	not	not	-
$A \cap B$	not	+	not	+

Table 5.2: The inclusion of faces of volumes A and B (+,–) into the result of Boolean operations

predicates are:

A_{loop} – non-distorted

B_{loop} – distorted

In case B there are edge concretes – one for each edge of the original loop. There are the same two predicates A,B for edge and surface patch– concepts like for loop concepts. A distorted edge is finally represented by a list of subedges. A distorted surface patch is represented by a list of *holes* (negative loops), that are inside the outer boundary loop.

5.3 The structure of the object model

5.3.1 The horizontal hierarchies

The knowledge representation mechanism provides two types of horizontal hierarchy of Object model concepts – **modality sets** and **specialization links** . Facts represented by modality sets could be represented either by the specialization links or by additional part-concepts as well. Due to modalities a compact knowledge base and a homogenenous analysis can be achieved.

We define tree–like variations of object classes with help of specializations (Fig. 5.7). Specializations allow limited variations of the object structure and of relative displacement of solids. It is intended that with the general concept of each object class most characteristic solids (probable with greatest dimensions) are related. Each more specialized concept contains more features (more parts) then its generalization.

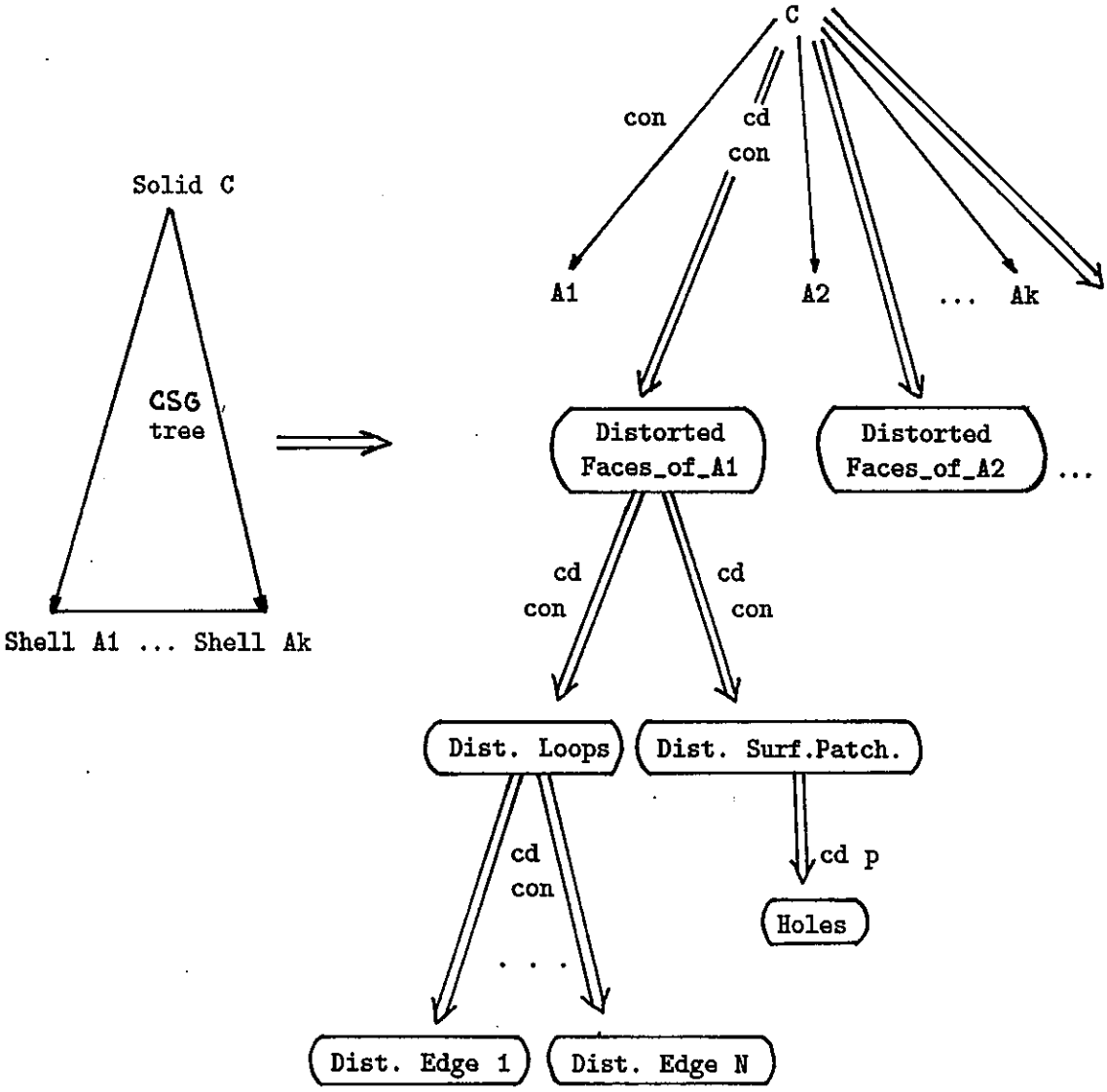


Figure 5.6: The boundary of a solid class

5.3.2 The knowledge for one description entity

Let some description entity **A** with substructures $K_1, ..., K_n$ be given in the model-scheme. One or more concepts of type A or type *hyp_A* can be provided in the Object Model (Fig. 5.8(a)). The substructures $K_1, ..., K_n$ are aggregated in each A (or *hyp_A*) concept into a finite set of **modalities**. The difference between A and *hyp_A* lies in their ancestors along the specialization link. In first case concepts for more specialized description entities of **A** are provided, whereas in the second case one specialization concept *verify_A* is provided first and the more specialized A's follow next.

Thus the analysis of information relevant to entity **A** proceeds in **two phases** (Fig. 5.8(b)). The **first phase** represents a *hypothesize-and-test an instance* – paradigm and deals with the instantiation of concept *hyp_A*. It is summarized by a sequence of **steps 1–4.**

- **STEP 1:** create a *partial instance of hyp_A – inp(hyp_A)* (the hypothesized instance)
  A partial instance of *hyp_A* is created by the application of RULE 1 to the instances of obligatory (context independent) links.

- **STEP 2 :** predict the context dependent links
  The instantiation of context dependent links is controlled be the top–down evaluated prediction, which arises from the inp(A). The context dependent instances are created after the predictions have been propagated downward by the application of RULE 5. These predictions are more restrictive if the concept can not be "occluded" in the scene. If a global verification of instances of A is made, these restrictions describe only the space boxes where the instances should be looked for. Examples:
  1) a volume instance can be evaluated on base of obligatory faces first and the surface description for these and remaining faces can be evaluated top-down from this volume next;
  2) a loop description is evaluated on base of partially visible edges and the full edge parts are computed top-down from the loop instance.

- **STEP 3 :** instantiate the context-dependent links
  The context dependent links are instantiated by applying RULE 1 and RULE 2. It is important to notice, that "null matches" of these links are allowed.

- **STEP 4 :** create a *complete* instance of *hyp_A – in(hyp_A)* (the locally tested hypothesis)
  The hypothesized and locally verified instance of hyp_A is created by applying RULE 2. For the refinement of its attribute evaluations all confirmed context dependent concretes can now be used additionally. The **inherent** concretes represent substructures of the model-scheme concept A, which are always completely hidden from given viewpoint (self-occlusion of A).

- **STEP 5 :** instantiate the optional links
  The optional links in current modality set are instantiated by applying RULE 1, 2, 3.

- **STEP 6 :** create an *extended* instance of *hyp_A*
  Extended instances of *hyp_A* are created by applying RULE 3 to *in(hyp_A)* and to the the subsets of instances of optional parts.

The STEPS 5,6 are usually performed after the goal concept has been instantiated.

The **second phase**, called **global verification** is expressed by the instantiation of the specialization concept *verify_A*. This phase is performed either immediately after the instantiation of *hyp_A* or is delayed to the moment after the current goal concept has been instantiated. The actions are summarized by **steps 7–11.**

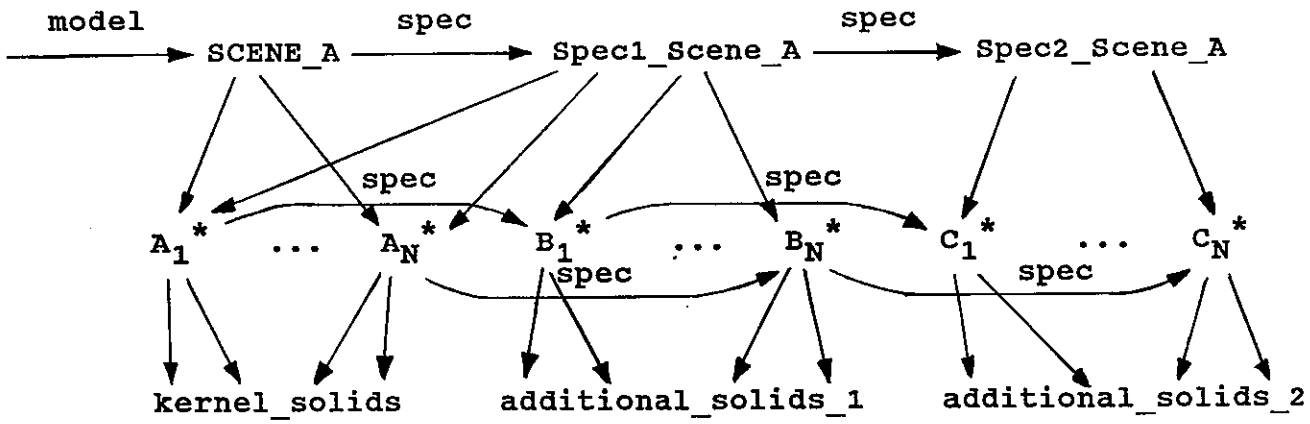


Figure 5.7: The structure of the object specializations

- **STEP 7 :** create the *partial* instance *inp(verify_A)*
  By applying the specialization rule an instance of *verify_A* is created out from the instance of *hyp_A*. The links K1,...,Km are inherited to the created instance. It is immediately a partial instance, because *verify_A* has no more context independent obligatory links.

- **STEP 8 :** verify *in(hyp_A)*
  For global verification a set of context dependent parts of *verify_A* is provided, which is compatible to the concrete set. These part-instances are top–down evaluated from the inp(A) and are classified into *confirmed* (if the bottom-up evaluated instance of concrete (the synthesized evaluation) is equal to the top-down prediction of the compatible part (derived evaluation)) or *non-confirmed* (if the synthesized instance is "smaller" than the derived one or it is a "null matched" one).

  The verification of the instance in(hyp_A) is expressed in terms of instantiations made for a subnet, that is linked to *verify_A* by context dependent part–links. These are are compatible to the non-confirmed parts. RULE 1, RULE 2 and RULE 1A are applied during this process. The first step of this process is the computation of the solid-class dependent distortions of each part of in(A). Next to the non-confirmed parts the visibility operations are applied. At end the achieved instances of optional parts are related to the concrete instances and are classified into *verified* or *non-verified.*

- **STEP 9 :** create a complete instance of *verify_A – in(verify_A)*
  The globally verified hypothesis of A is created by applying RULE 2 to the partial instance *inp(verify_A)* and to the *verified or non-verified* – part instances.

- **STEP 10 :** extended match
  Optional parts, one for each non–verified and non–confirmed instance of cd part are instantiated. These parts are matched directly with the remaining segmentation data, by applying a simple RULE 1.

- **STEP 11 :** create *extended* instances of *verify_A* – $\{in_k(A)\}$ (the verified hypotheses)
  Extended instances of *verify_A* are created by applying RULE 3 to *in(verify_A)* and to the instances of optional parts.

Just like for STEPS 5,6, the STEPS 10,11 are usually performed after the goal concept has been instantiated.

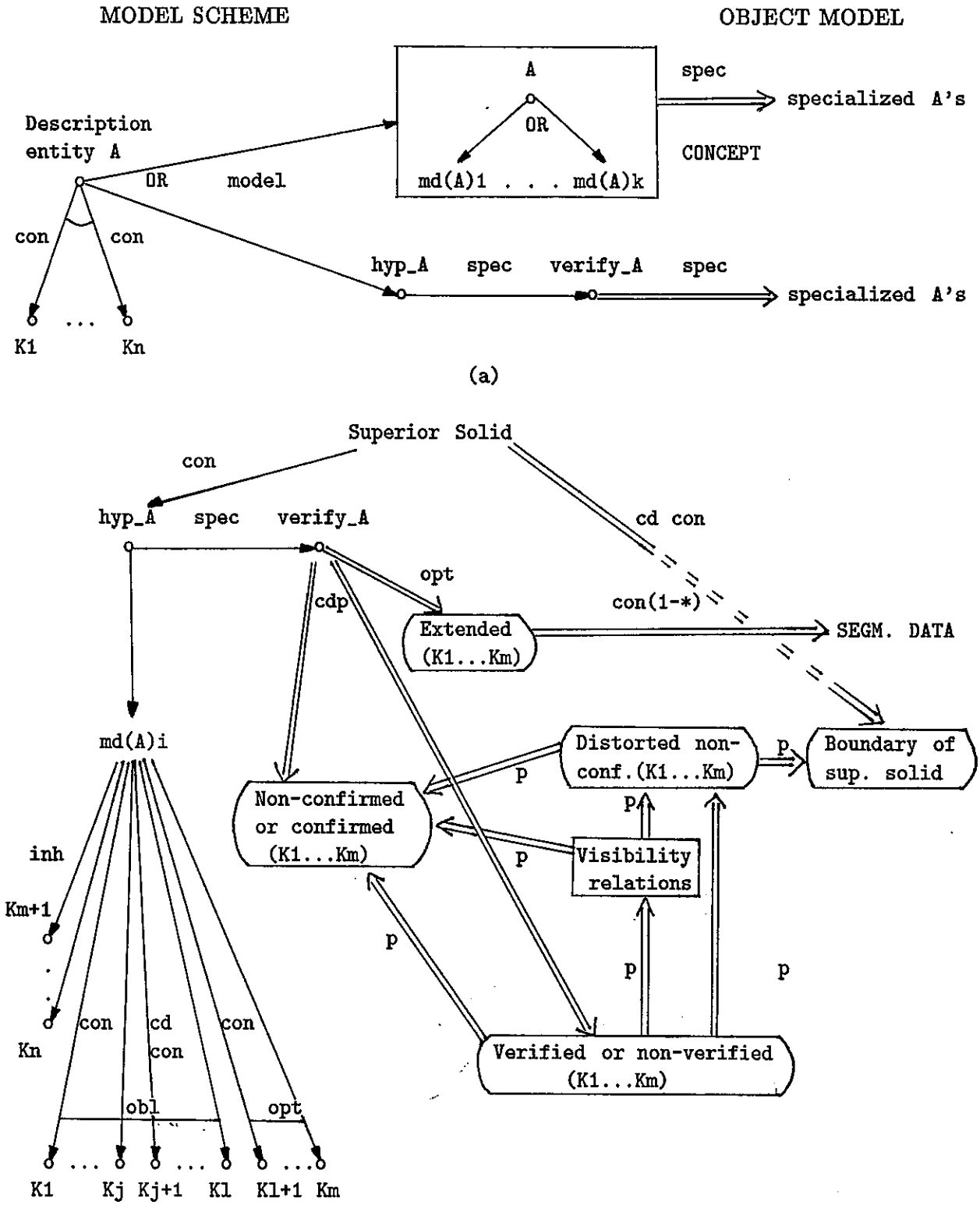
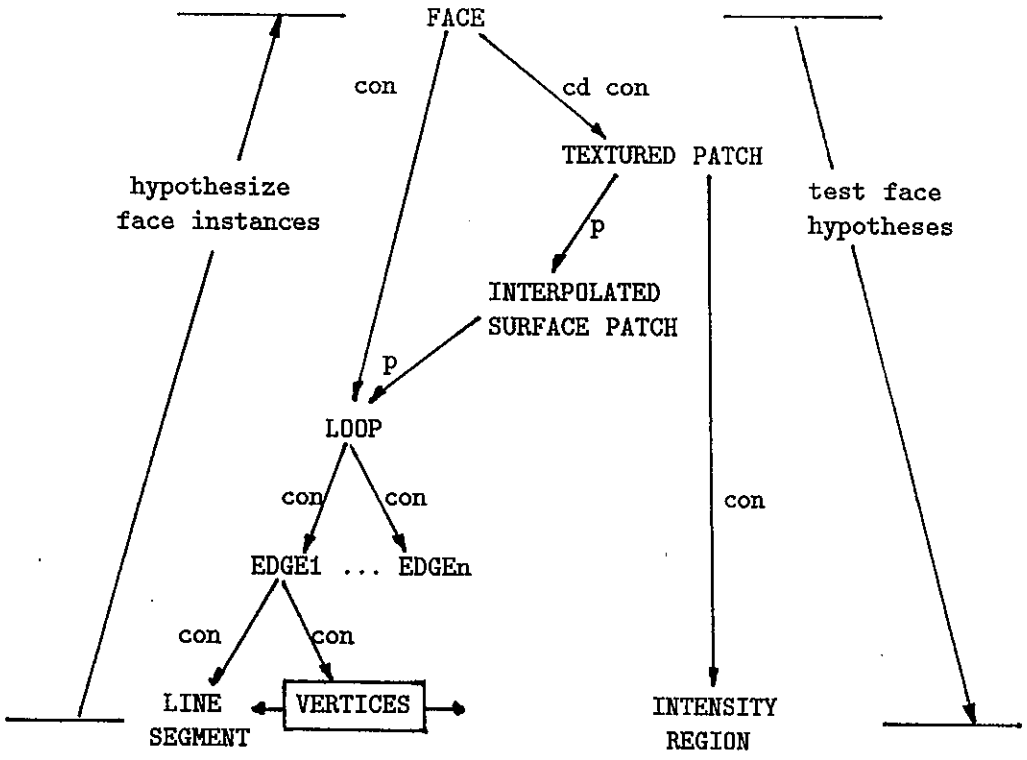
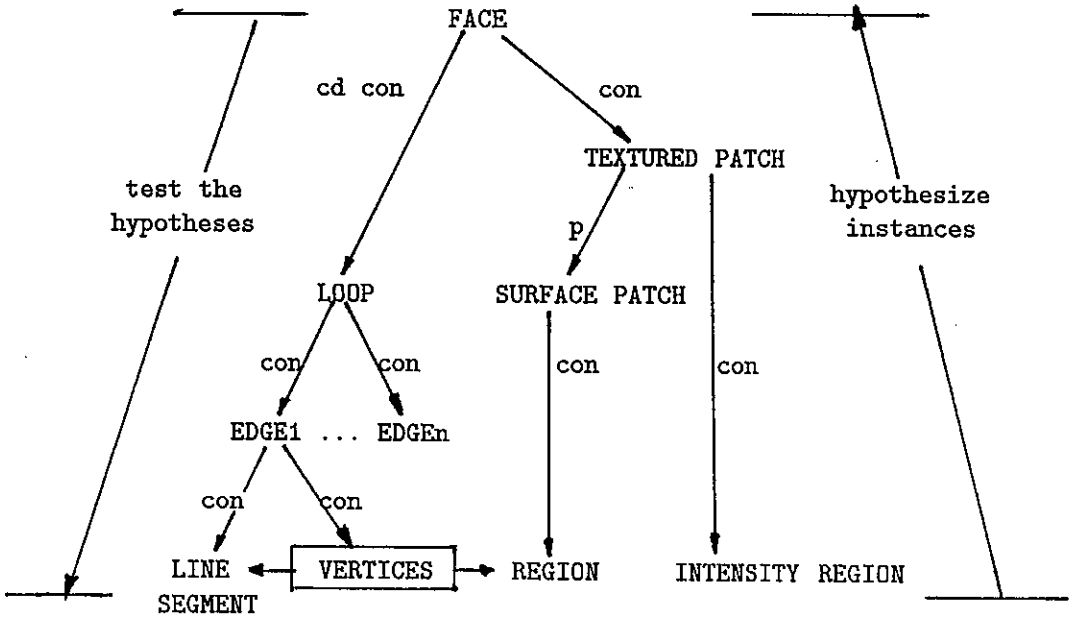


Figure 5.8: The knowledge for the instantiation of one concept



(a)



(b)

Figure 5.9: Duality of face modalities for: (a) sparse, (b) dense range images

5.4 The object model data

5.4.1 Knowledge for the hypothesize–and–test paradigm

By the distinction of context–dependent and context–independent links in a modality set, the *hypothesize and test*–paradigm can be simulated.

- *Modalities of edges and surface patches*
  By using R–concepts we can obtain line segments and junctions from pairs of adjacent regions. The line segment can represent an occluding or inner edge, whereas a junction stands for a real or virtual vertex. These elements are represented by R-concepts. Generally each edge class has five modality sets according to following occlusion– and imperfect segmentation–cases: fully visible, one side hidden, one end detected, two sides hidden, isolated.

  For dense range data the 3–D regions can be classified by the use of two predicates: "inner" and "occluding".

- *Modalities of loops*
  Full loops which belong to basic face types are hypothesized on base of partly visible edges only.

- *Modalities of faces*
  The performance of lower levels is distinct for sparse and dense range data (Fig. 5.9). In first case the probable hypotheses about face instances are created on base of boundary loops. The context-dependent concretes are here the interpolated surface patches. If these surface patches are matched, a refined face hypothesis can be obtained. For dense range data a week face hypothesis is created on base of a surface patch first. There is a constrained search for boundary elements next. After some of the were found in the image, a more complete hypothesis can be computed.

- *Modalities for shell and solid hypotheses*
  This is a set of viewpoint–dependent shell and solid representations. As long as we deal with views represented in 3–D space, the geometry of the shell or solid remains unchanged (Fig. 5.10, 5.11).

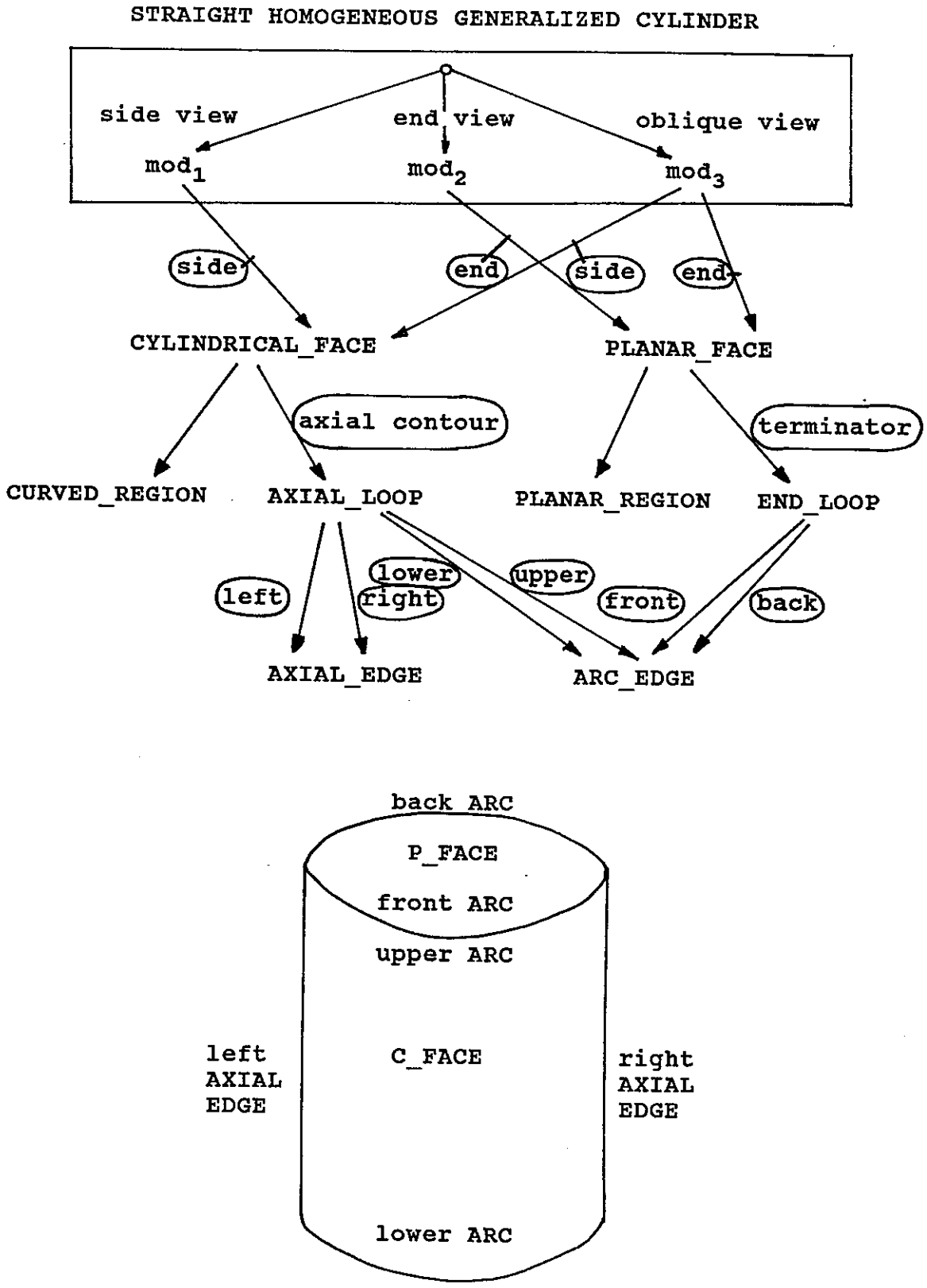


Figure 5.10: Example of shell modalities

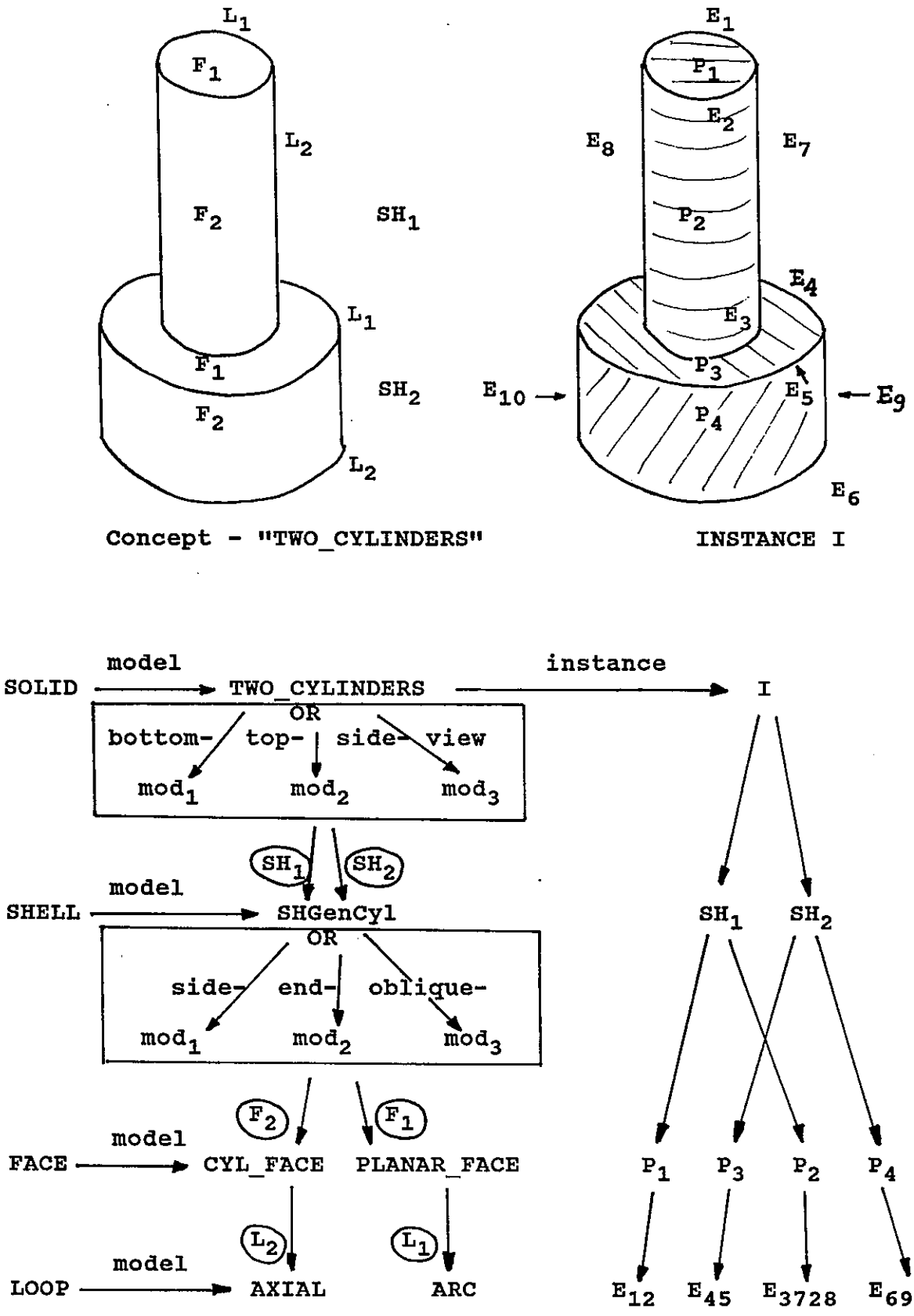


Figure 5.11: Examples (a) of a solid model and (b) of a solid instance

5.4.2 Knowledge for the verification process

The verification process starts after a complete instance I(G) of the goal concept G has been created. Three types of operations can be performed during the verification of a particular instance I(A):

gluing, visibility, verifying

First the solid-dependent distortions of instances of non-confirmed parts are computed (the gluing operation). The type of distortion is described by context dependent links of a superior solid concept.

So called visibility R–concepts are applied next to such obtained instances and to the confirmed instances. These R–concepts are unlimited parts of the so called verified & non-verified optional parts of A. A visibility R-concept is classified into *terminal* or *non-terminal*. The existence of terminal visibility instances results in a partly or full reduction of the part instances – the instances of (optional) verified & non-verified parts are created. The non–terminal R–concepts are contexts of visibilty concepts on lower levels (Fig. 5.12). Hence the visibility relations between objects control the computations of such relations between solids; relations between solids control the inter–shell computations and so on. The "reduced" part instances of in(A) are related to the synthesized concretes of in(A) (verifying). An optional part is labeled as verified if it is similar to its compatible concrete. If the concrete instance is "smaller" than the part instance, then the latter is labeled as non-verified. A confirmed part should not be reduced during the visibilty step.

The distortions introduced by gluing operations may be as follows:

G_{shell} – elimination of faces
$G_{surfacepatch}$ – making holes in surface patches
G_{loop} – changing the boundary of loops.

The terminal visibility operations detect following facts:

V_{object} – fully hidden solids
V_{solid} – fully hidden volumes
V_{shell} – fully hidden and back faces
$V_{surfacepatch}$ – hidden holes (or parts of holes)
V_{loop} – partly hidden edges

Following techniques are applied in these visibility operations :

$T_{object,solid}$ – intersection check of projected bounding boxes
T_{shell} – test of the surface normal direction (additionaly to the previous test)
$T_{surfacepatch,loop}$ – edge clipping technique

5.4.3 The extended match

If a non-verified instance exists one tries to explain the difference between the hypothesis generation- and verification-steps by mistakes of the segmentation process or by the discrete character of the imaging process. The assumed mistakes are: 1) non–merged similar line segments or regions, and 2) non–detected segments.
The other faults – merged non-similar segments and detected imaginery segments – are handled by the matching strategy itself, due to the model-to-image matching process and the partial matching.

The created extended instance of A will contain instances of additional optional parts for each non-verified part instance. This optional instance is the result of a direct match between the predicted part and the remaining (non-explained) image segments.

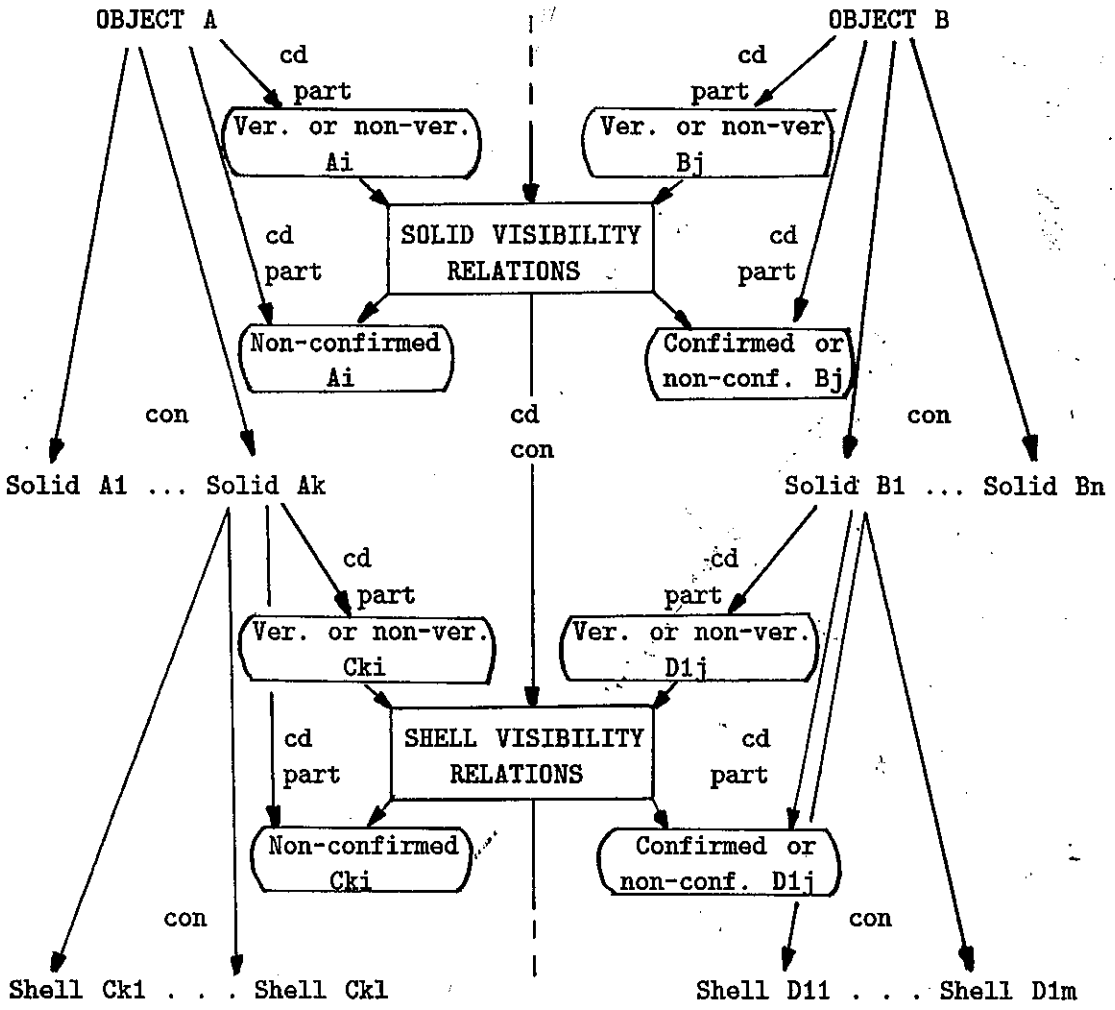


Figure 5.12: The hierarchy of visibility operations

- *Edge parts of a loop*
  The "visible" parts of edges, which were not reduced to edge concrete instances, are matched with the non-explained line segments. The overlapp relation between the edge part instance and these line segments is tested. In this way a segmentation fault – the existence of artificial vertices caused by non–merged line segments – can be corrected (Fig. 5.13(a)).

- *Surface patch parts of a face*
  A surface-patch is matched with regions in the segmentation data. One type of a segmentation mistake – the existence of (conceptually) similar non–merged regions – can be corrected (Fig. 5.13(b)).

- *Volume parts of a solid, solid parts of an object*
  Volumes that are far away from the viewer can not be detected in 3–D terms, because their boundary sizes are below the imaging threshold. Hence a volume (shell) should be matched with remaining image regions (Fig. 5.13(c)). Similarly the expected derived solid parts are matched with the image regions.

5.5 The attribute tables data

The attribute tables contain such properties of concepts from the object model, which can be extracted directly from the low–level image description. After the attribute tables have been filled with relevant image data, they are used like indices of concepts from the model.

- *Object indices*
  By a simple threshold along the z axis, the division into indoor and outdoor pixels is done. The outdoor pixels are grouped by 2-D techniques into regions. The physical and textural features of outdoor regions can be used to select outdoor objects in the model.

- *Solid indices*
  Numerical and textural features of background indoor (3-D) regions are used to select background solids.

- *Volume indices*
  Central solids are represented in terms of 3-D volumes. A volume in terms of primitives is a connected "blob" of 3–D regions, which can be supported by a supporting plane below. The dimensional features of the extracted blobs are for example orientation of the most elongated bounding rectangle on the ground projection image of the blob, the side view projection image onto a plane defined by the orientation and the z-axis. Some invariant features (i.e. 2–D moments, Fourier coefficients) of the retained silhouette or symmetricity features of projected blob can be used as cues for volume detection. For special volumes (i.e. generalized cylinders, which have a regular side projection) pairs of parallel major bounding boundaries constitute the cues for detection. Concave or convex inner edges determine the classes of volumes.

- *Face indices*
  The line segment classes of inner boundary determine the classes of face pairs.

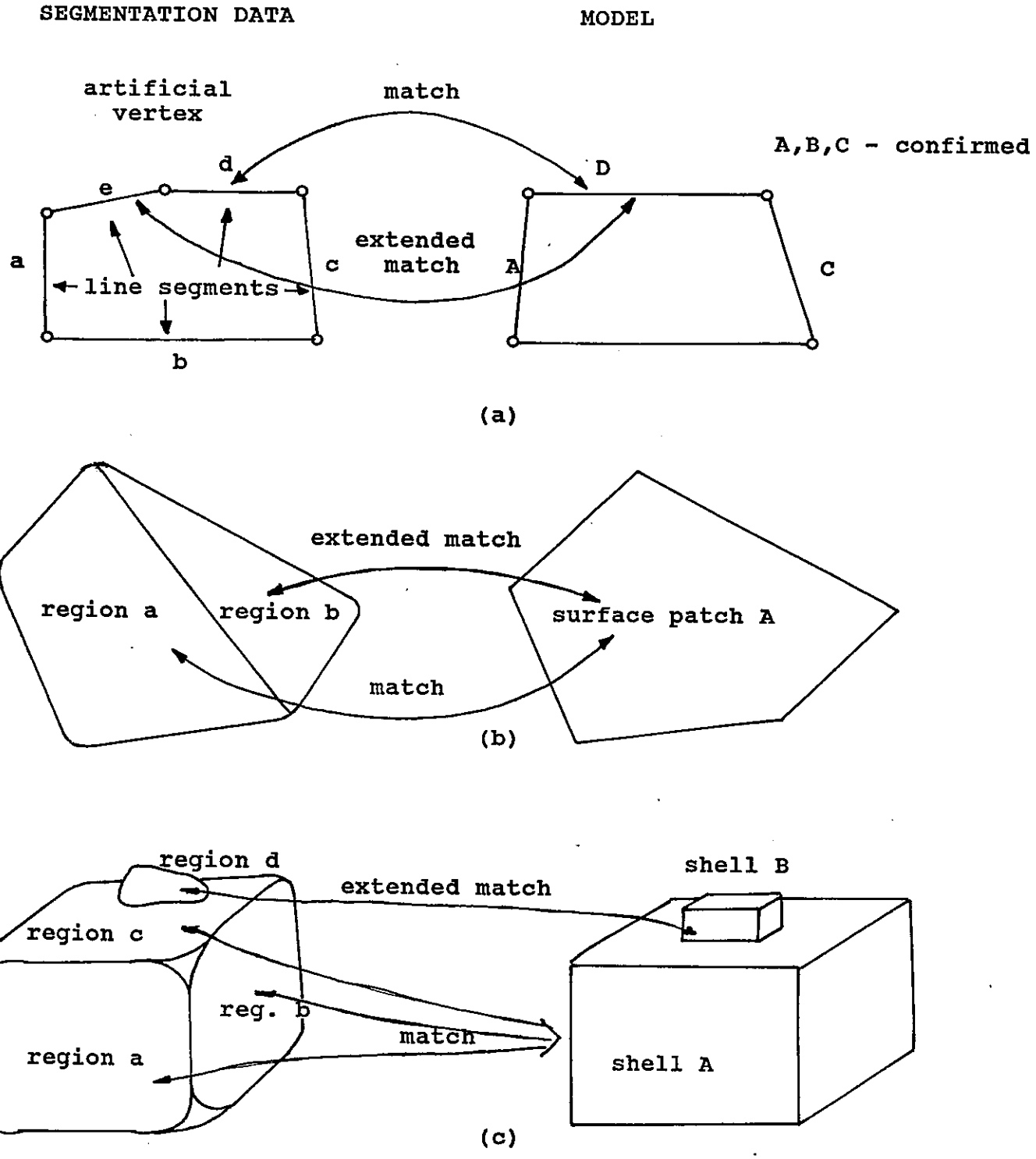


Figure 5.13: Extended match of (a) edges, (b) surface patches, (c) shells

Chapter 6

Recognition strategy

In this chapter the three elementary search problems are stated and two control algorithms are proposed – the *extended bi–directional* control and the *bi–driven* one. The term *object* means here an element from the *instantiation list* of a *goal concept* and can be a concept, a modified concept or a partial instance.

6.1 The search problems

There are three elementary search problems:

- state space search for an actual goal concept (level (1)– nodes)
- AND/OR graph search for the set of instantiation lists
- state space search for best instantiations (level (2)– and (3)–nodes)

The AND/OR graph search is performed either outside the state space (in the extended bi–directional control) or the goal concept expansions are integrated into the state space search (like in the bi–driven control).
The two remaining search problems are solved in the same state space. Candidate nodes for expansion are selected by the A^*-tree search algorithm. The selected node can be either on level (1), (2) or (3). The embedding of the three search problems into the bi–directional control is shown in Fig. 6.1.

6.1.1 Problem 1: (modified) goal concepts

This space consists of the model concepts referred in the attribute tables and of their descendants along the "concrete of" and "specialization" links. Its graphical representation is an OR graph (Fig. 6.2). The overall goal is to find the optimal path between one of the initial concepts and one of the terminal concept. The optimality is a function of the match between the goal concept and the segmentation data. Each created successor goal contains the match of its predecessor goal concept to the segmentation data. This match is part of its own match (if a more abstract goal is selected) or can be partly modified (if a more specialized goal is selected).

6.1.2 Problem 2: competitive modality sets

In the bi–directional control a two–step instantiation is assumed – first a complete model expansion is performed and the competitive instantiation lists are found, and then the instantiations for objects from given list are made. The expansion is performed on two levels: the obligatory or optional links are expanded. In first case RULE 1 and RULE 2 are considered only. In the second case the RULE 3 for given goal concept and the RULE 1 and RULE 2 for the subconcepts of the goal are considered. As it was shown in chapter 3 the expansion of the goal results in an AND/OR graph.

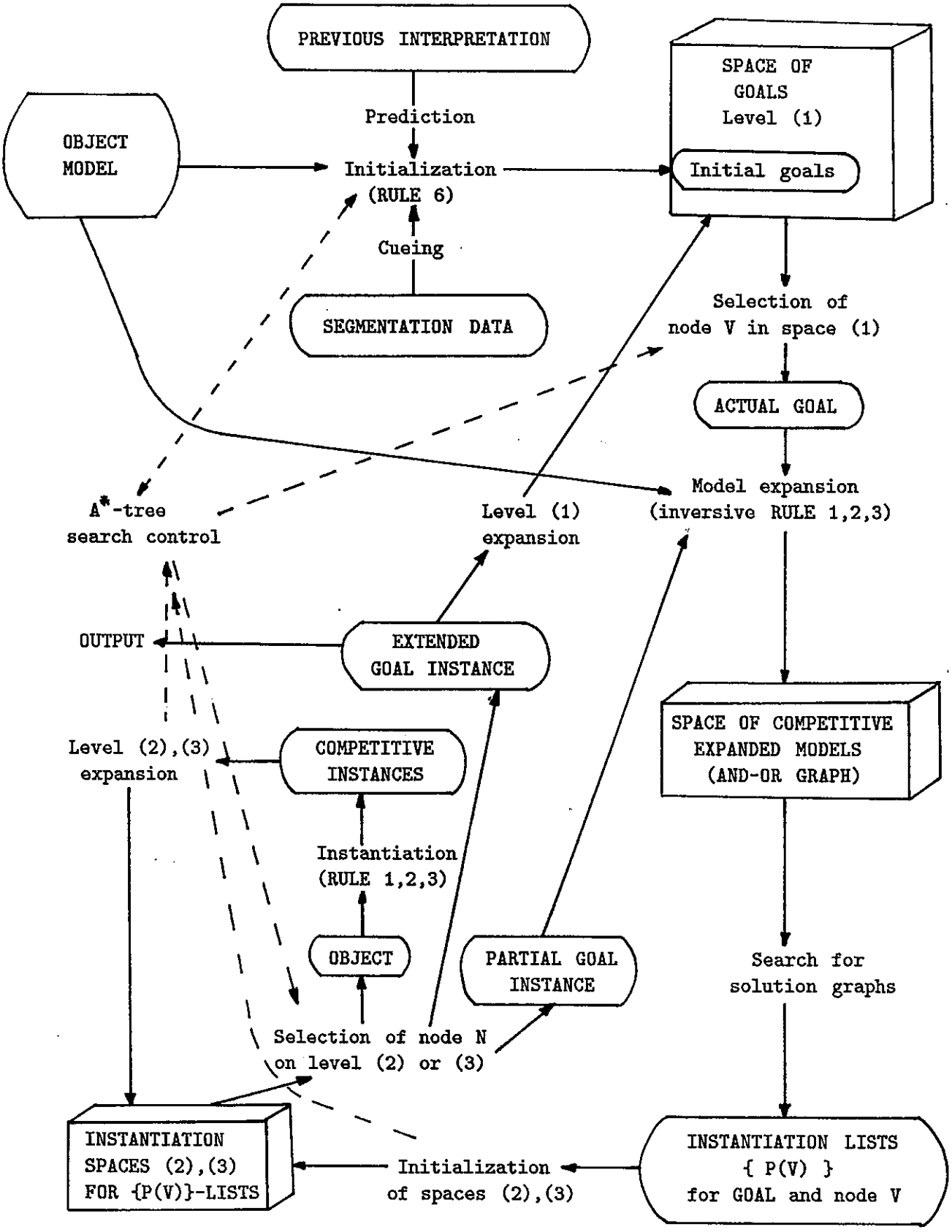


Figure 6.1: The three search problems

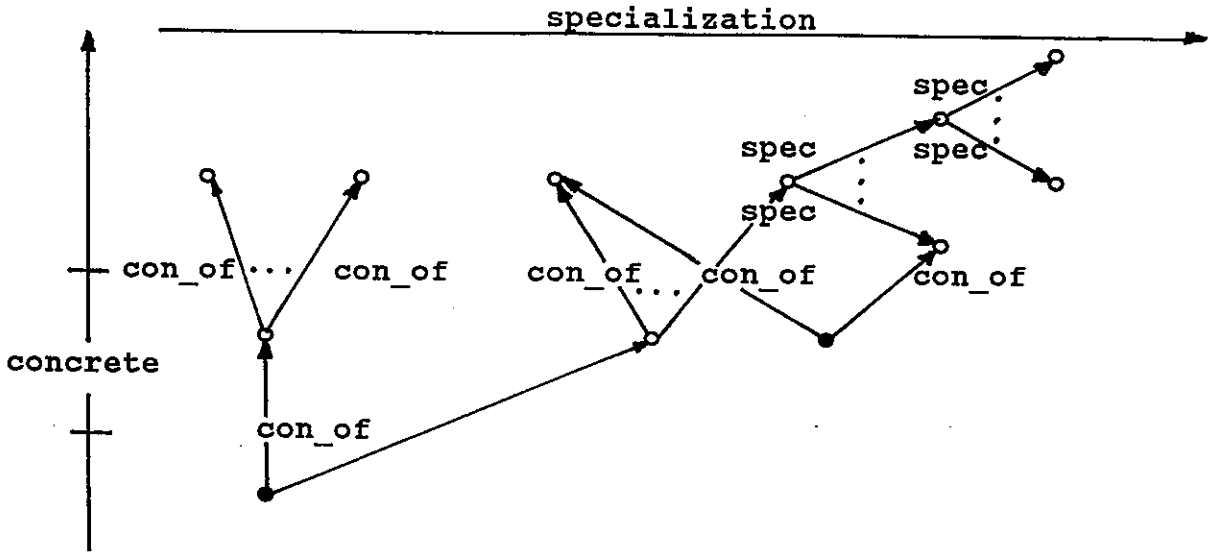


Figure 6.2: The space of goal concepts

6.1.3 Problem 3: competitive instances of one concept

Search nodes for objects from the obligatory (level (2)–nodes) or optional (level (3)–nodes) premises of the goal concept are expanded if competitive instances of given object exists. These multiple instances are results of different attribute evaluations in RULE 1 and RULE 2 or they result from different optional premises in the RULE 3.

6.1.4 Extensions of the bi–directional control

The basic control is first extended into so called *extended bi–directional* control by following elements:

- extending the incremental model exploration due to *specialization* links
- further structuring the model along the part of–hierarchy in order to focus on important aspects during the analysis:
  - distinction of *modality* sets in RULE 1 and RULE 2
  - dividing the modality sets into *obligatory* and *optional* (applying RULE 3)

Finaly the *bi–driven* control is achieved by incorporating additionally above extensions the data–driven elements:

- overcoming the limitations of the model–to–image matching:
  - *global* relations (R–concepts and RULE 1A)
  - *iterative/unlimited* parts (an iterative model–to–image match)
- constraint propagation:
  - mixed goal expansion and instantiation

- applying RULE 4 and 5 for expanded model modification

- search space pruning tools:
  - limitation of successors
  - staged search

6.2 Extended bi–directional control

The extended bi–directional control is proposed in Table 6.1.

6.2.1 Concept recognition– and inheritance–inferences in model network

Until now the model was incrementaly explored by using the first type of inference – the *concept recognition* along the *part–of* hierarchy. Now we utilize the second type of inference also – the *inheritance of conceptual attributes* along the *is–a* –hierarchy.
The search for an optimal path in an OR graph of (modified) goal concepts is represented by the expansion of *goal concept*–nodes (search level (1)). These nodes are expanded if competitive successor goal concepts exist, which are:

- initial goals (by applying RULE 6 to the attribute tables) or bottom–up modified superior concepts (by applying RULE 4 to the instance of actual goal)
- specialization concepts of actual goal instance on the most abstract level of the concrete hierarchy

Each goal concept node is the root node of an own sub–space of *obligatory description*–nodes (search level (2)). If no modalities and RULE 1 and 2 are considered only, the actual instantiation list P(V) for given goal is computed top–down with respect to the actual space node V (to the actual instance set INST(V)). The search level (2) is expanded for competitive instances of a concept from P(V).

6.2.2 Modality sets

Due to the *modalities* of concepts the problem of search for all solution graphs (instantiation lists) in an AND–OR graph (expanded model) M for current goal G(V) arises. The solution of this problem is performed outside the state space.
The model network is prepared for analysis if an initial expanded model is pre–computed for each concept of the model network. The expanded model is abstractly an AND–OR graph. Hence the competitive instantiation lists are solution graphs of an AND–OR graph. The actual instantiation lists { P(V)} for given goal concept are computed during search with respect to the actual space node V (to actual instances from INST(V)). The set { P(V) } corresponds directly to the pre–computed solution graphs (if INST(V)$\doteq$ 0) or is a a subset of them (if sub–concepts are already instantiated). After that, successors of the root node of each level–(2) subspace are created – one for each competitive instantiation list in P(V).

6.2.3 Obligatory and optional links – RULE 3

Due to the optional parts needed by the RULE 3, the *optional description*–nodes (on the search level (3)) must be introduced. Before an *extended* instance can be created out from the complete one a search for instances of optional concepts from OPT(N) is performed, which is a *homomorphic* expanded model–to–image match (this means that the same segmentation data set is available for instances of all optional parts of given concept). The competition of instances of optional concepts is solved by the RULE 3, which determines consistent subsets of optional instances.

Input: segmentation results SEGM, filled attribute tables, set CG of competitive goal concepts

Initialize: search tree $S = (V, E), V = \{R\}, E = \emptyset$; lists OPEN $= \emptyset$, CLOSED $= R$

FOR all concepts K in CG DO: apply RULE 6 to K

FOR each mod(K) generated by RULE 6 DO:

- activate function *MODEL-EXPANSION*(mod(K),R, M)
- FOR each solution graph K_i of M DO:
  - generate one successor node V_i of root R in search tree S; add V_i to OPEN
    generate the instantiation list P(V_i) for the solution graph K_i
    $h(V_i)$ = the judgement of mod(K) with respect to K_i
    make INST(V_i), GOAL2(V_i) and OPT(V_i) empty; GOAL1(V_i) = K
    PRIM(V_i)= SEGM

WHILE OPEN is not empty DO:

- select the node N with best score h from OPEN
- IF P(N) is not empty
  THEN choose the last object C on P(N)
  - IF COMPLETE(C)= T
    THEN activate function *OPTIONAL-MODEL-EXPANSION*(C,N,M)
    - FOR each instantiation list P_i in model M DO:
      - generate one successor node O_i of N in S; add O_i to OPEN
        P(O_i)= P_i; $h(O_i)$ = judgement($h(N), P_i$); INST(O_i)= INST(N)
        GOAL1(O_i)= GOAL1(N); PRIM(O_i)= PRIM(N)
        GOAL2(O_i)= N; OPT(O_i)= { optional links of C}
    - ELSE generate instances of C by using RULE 1, RULE 2 or RULE 3
    - FOR each instance I(C) of C DO:
      - generate one successor O of N in S and OPEN; P(O)= P(N) – C
        INST(O)= INST(N)+I(C); h(O)= judgement(I(C), h(N), P(O))
      - IF RULE 3 was applied
        THEN remove all nonused I(K) for $K \in$OPT(N) from INST(O)
      - GOALi(O)= GOALi(N) i=1,2; OPT(O)= OPT(N)
      - IF GOAL2(V) is not empty (space (3)) AND C $\in$OPT(O)
        THEN PRIM(O) = PRIM(GOAL2(O)); remove C from OPT(O)
        ELSE PRIM(O) = PRIM(N) – { data covered by I(C)}
  - remove node N from OPEN and add it to CLOSED
- ELSE IF GOAL2(N) is not empty (return from level (3))
  THEN take P(N), OPT(N), GOAL2(N) from node GOAL2(N)
  COMPLETE(C)= F, where C is the last object in $P(O)$
  - ELSE IF specified topological degree has not been achieved
    THEN determine a set B of superior objects of I(GOAL1(N)) (RULE 4)
    ELSE send the node N to the OUTPUT (PARTIAL SOLUTION)
    determine a set B of specialization objects of I(GOAL1(N))
  - IF the User decides that an analysis goal has been reached THEN STOP
  - FOR each modified concept mod(K) in set B DO:
    - activate function *MODEL-EXPANSION(mod(K),N, M)*
    - FOR each solution graph K_i in M DO:
      - generate one successor V_i of N in S; add V_i to OPEN
        generate P(V_i), INST(V_i) and PRIM(V_i) for K_i, V_i;
        make GOAL2(V_i) and OPT(V_i) empty
        $h(V_i)$ = judgement(mod(K)); GOAL1(V_i)= mod(K)
  - remove node N from OPEN and add it to CLOSED

STOP – unsuccessful end of analysis

Table 6.1: The extended bi-directional control

6.2.4 Model expansion and creation of instantiation lists

In Table 6.2 the model expansion step for a goal object K of a space node V is outlined. During the analysis the initial solution graph set is modified by cutting the subgraphs, which are already instantiated.

common list IDENT for the identification slots MODEL-EXPANSION(goal K, space node V, expanded model M):	
place K at the list M	
open a list M(K)	
IF	K is *identified* to be already expanded
THEN	MERGE both K exemplars and RETURN
IF	K is a R–concept
THEN	RETURN
open a list M(K); attach M(K) to K by an OR arc save the identification slots of K in IDENT	
FOR all modalities of K – $md_i(K)$ DO:	
place K^i at the end of list M(K) open a list P(K_i) for $md_i(K)$	
FOR	all concepts A (which are obligatory context independent links in $md_i(K)$ AND (an instance of A is not yet related to K) DO:
	MODEL-EXPANSION(A, V, P(K_i)))
open a list E(K_i) place $K^i_{partial}$ at the list E(K_i) attach the list P(K_i) to $K^i_{partial}$ by an AND arc	
FOR	all concepts A (which are obligatory context dependent links in $md_i(K)$) AND (an instance of A is not yet related to K) DO:
	MODEL-EXPANSION(A, V, E(K_i))
attach the list E(K_i) to K^i by an AND arc	

Table 6.2: The obligatory model–expansion procedure

The next step is to find a set V_j of actual solution graphs (or by linearizing them – a set of solution lists is obtained). At end each instantiation list P_j is created out from a solution list by adding to its top a set of extended objects { K^i_{ext} } – one for each complete object K^i from the instantiation list, which has optional parts due to modality $md_i(K)$. To each object { K^i_{ext} } the predicate COMPLETE({ K^i_{ext} })= T is related, whereas for all other objects K the predicate COMPLETE(K)= F is specified. During the analysis the RULE 3 can be applied to K^i_{ext} only if the optional links have been instantiated. Up to that moment COMPLETE({ K^i_{ext} })= T. The optional model expansion step is performed as follows:

OPTIONAL-MODEL-EXPANSION(goal K^i_{ext}, space node V):	
open a list M	
FOR	all optional links C of K due to $mod^i(K)$ DO:
	MODEL-EXPANSION(C,V,M)
attach M to K^i_{ext} by an AND arc	

The instantiation lists are computed in the same way like in obligatory expansion. A next–level instantiations of each optional part (in a subordinate level–(3)–space)) are considered.

6.3 The bi–driven control

The final bi–driven control is presented in the tables 6.3 – 6.4. The input data and the initialization are the same as in Table 6.1 (and are omitted in Table 6.4).

6.3.1 R–concepts for generic relations and RULE 1A

So called *R–concepts* are represented by the same data structure as normal concepts, but they are instantiated by a modified RULE 1A. A R–concept is not expanded – there are not made distinct objects for parts of a R-concept. Each instance tuple of concrete– or part– concepts of given R–concept is the premise for instantiation of this concept.

6.3.2 Unlimited links and iterative search sub–spaces

If some optional link provides a special marker "*", then it will be searched for an image–dependent number of instances of a concept refered by this link. After an instance of an optional link C is found, the search space node N is expanded by two types of nodes: node O from which the next concept from the instantiation list is instantiated, and node O^1 from which the instantiation of optional link is iterated but with restricted DATA set. The process of iteration is controlled by the set PRIM, which refers to the image data already covered by the INST set. PRIM(O^1) containes only these segmentation results, which are not interpreted by instances from INST(N) yet, whereas PRIM(O) refers the situation when the actual level (3) was started.

6.3.3 Constraint propagation

Until now, while selecting an instance we do not check the instances selected previously on the path through the search space and did not know if the set of instances would constitute a geometrically consistent (partial) description. This causes very often an exploration of mainly whole search space. An early detection and pruning of the inconsistent paths is of great importance. In general the structural relation is tested only after the superior concept has been instantiated. In the same step the newly introduced instance can constrain the later instantiated parts and the superior concept.
Modified concepts in the instantiation list
In order to propagate constraints from already created instances the instantiation list should contain modified concepts. If the goal is already a modified concept, restricted modified concepts for parts and concretes can be created by the application of the RULE 5. During search after instantiating some element from the instantiation list, the constraints are propagated first bottom-up by the use of RULE 4 to all superior partial entitites from the instantiation list (updating the superior modified concepts for partial instantiated concepts) and then top–down by applying RULE 5 to context–dependent parts and concretes (from the instantiation list) of the updated modified concepts.
Mixed expansion and instantiation
In the extended bi–directional algorithm the successors of the root nodes of local subspaces (2) or (3) are created for each competitive instantiation list. Practicaly instead of one local root node there are multiple local roots (one for each instantiation path). Hence the search space is growing and there is not an easy judgement scheme for the instantiation lists of given concept.
The idea is to integrate the goal expansion and AND/OR search into the instantiation space. Because the instantiation list is assumed to contain modified concepts, the constraints propagated from previous instantiations will reduce the modality sets applicable in given situation.

6.3.4 Pruning tools

By the use of the pruning tools there is now no guarantee, that the optimal path will be found.

Limitation of successors
Immediately after instantiation almost all created instances are accepted except a few having the smallest judgement value.

Staged search
After returning from a level (3)–node or from a level (2)–node some subset of nodes in the leaved subspace having a small judgement is removed form the OPEN set.

MODEL-EXPANSION-STEP(object mod(K), space node N):		
FOR all modalities of the concept K – K^{md_i}, that satisfy the constraints of object mod(K) DO:		
generate one successor node O of N in OPEN; P(O)= P(N) $h(O)$ = judgement($h(N)$, priority(K^{md_i}))		
IF	there are optional links in K^{md_i}	
THEN	place $mod(K)^i_{ext}$ at the top of list P(O)	
place $mod(K)^i$ at the end of list P(O)		
FOR all concepts A (which are obligatory context dependent links in K^{md_i}) AND (an instance of A is not yet related to K) DO:		
	create mod(A) by applying RULE 5 to mod(K) due to modality K^{md_i}	
	IF	mod(A) is *identified* to be already on the list P(O)
	THEN	MERGE both mod(A) exemplars
	ELSE	add mod(A) at the end of P(O)
place $mod(K)^i_{partial}$ at the end of list P(O)		
FOR all concepts A (which are obligatory context independent links in K^{md_i} AND (an instance of A is not yet related to mod(K)) DO:		
	create mod(A) by applying RULE 5 to mod(K) due to modality K^{md_i}	
	IF	mod(A) is *identified* to be already on the list P(O)
	THEN	MERGE both mod(A) exemplars
	ELSE	add mod(A) at the end of P(O)

Table 6.3: The model–expansion step

FOR each mod(K) generated by RULE 6 DO:
generate one successor node V of root R in search tree S; add V to OPEN
P(V)= { mod(K) }; h(V)= judgement(mod(K))
make INST(V), GOAL2(V) and OPT(V) empty; GOAL1(V) = mod(K); PRIM(V)= SEGM

WHILE OPEN is not empty DO:
select the node N with best score from OPEN

IF P(N) is not empty
THEN choose the last object C on P(N)

IF C is an object of a R–concept
THEN generate instances of C by using RULE 1A; add instances of C to INST(N)
remove C from P(N); h(N)= judgement(INST(N), h(N))

IF COMPLETE(C)= T (C is an extended object)
THEN generate one successor node O_i of N in S; add O_i to OPEN
$h(O_i) = h$(N); OPT(O_i)= { optional links of C} ;
INST(O_i)= INST(N); GOALi(O_i)= GOALi(N) i=1,2; PRIM(O_i)= ~~DATA~~ PRIM(N)
P(O_i)= OPT(O_i); remove node N from OPEN, add it to CLOSED

ELSE IF the instantiation premise for C is satisfied
THEN generate instances of C by using RULE 1, RULE 2 or RULE 3
FOR each WELL SCORED instance I(C) of C DO:
generate one successor node O of N in S; add O to OPEN
INST(O)= INST(N) + I(C); h(O)= judgement(INST(O), h(N))

IF RULE 3 was applied
THEN remove all nonused I(K) for K ∈OPT(N) from INST(O)

P(O)= P(N) – C; GOALi(O)= GOALi(N) i=1,2
OPT(O)= OPT(N); MODIFY P(O) by applying RULE 4 to I(C)

IF GOAL2(V) is not empty (space (3)) AND C ∈ OPT(O)
THEN PRIM(O)= PRIM(GOAL2(O)); remove C from OPT(O)

IF C is an iterative object C_i
THEN generate one successor O^1 of N in S and OPEN
P(O^1) = P(N) ∪ { C_{i+1} })
INST(O^1)= INST(O); GOALi(O^1)= GOALi(O)
PRIM(O^1)= PRIM(N) – {data covered by I(C)}
OPT(O^1)= OPT(N); h(O^1)= judge($h(N)$,C_{i+1})

ELSE PRIM(O) = PRIM(N) – { data covered by I(C)}

ELSE activate function *MODEL-EXPANSION-STEP*

remove node N from OPEN, add it to CLOSED

ELSE IF *GOAL2(N)* is not empty (return from level (3))
THEN take P(N), OPT(N), GOAL2(N) from node GOAL2(N)
remove from OPEN each V if GOAL2(V)==GOAL2(N) & h(V)< h(N)– θ_2
COMPLETE(C)= F , where C is the last object in P(N)

ELSE IF specified topological degree has not been achieved
THEN determine a set B of superior objects of I(GOAL1(N)) (RULE 4)
ELSE send the node N to the OUTPUT (*PARTIAL Description*)
determine a set B of specialization objects of I(GOAL1(N))

IF the User decides that an analysis goal has been reached THEN STOP

FOR each modified concept mod(K) in set B DO:
generate one successor node V of N in S; add V to OPEN
P(V)= { mod(K)}; generate INST(V), PRIM(V) for V and mod(K)
make GOAL2(V_i) and OPT(V_i) empty
h(V_i)= judgement(mod(K)); GOAL1(V_i)= mod(K)

remove from OPEN each V if GOAL1(V)== GOAL1(N) & h(V)< h(N)– θ_1
remove node N from OPEN; add it to CLOSED

STOP – unsuccessful end of analysis

Table 6.4: The bi–driven control

6.4 Judgement of the search space nodes

All the search space nodes are judged with respect to the overall goal, which is the generation of a scene description that best covers the (significant part of the) segmentation data. The comparison of nodes in the global search space is defined by the lexical order of a four–element **judgement vector**:

[quality, certainty, priority, primitives/covering]

The judgement vector allows the system strategy to focus on promising partial descriptions. The calculation of the judgement vector depends from the type of node. For level–(1) node it is equal to the judgement of its modified goal. The judgement of a level–(2) node depends from its INST set and the instantiation path of actual goal GOAL1. The judgement of a level–(3) node O is a function of the scores of level–(2) instances INST(GOAL2(O)) and of the instantiation path and scores of level–(3) instances.

The first and most important components are the **quality scores**, which refer not only to the actual instances but also reflect the chance of extending the partial description to a complete one, which covers the main part of the input data. An optimistic estimate of the quality achievable by the extension of the partial description is given as the sum of the quality of the space node content and an upper bound of the qualities of descriptions covering the remaining part of the input data.

Judgement vector of level (1)–node V:

- $Quality_G$ – quality of the (modified) goal GOAL1(V)
- $Certainty_G$ – certainty of the (modified) goal
- $Priority_G$ – priority of the goal concept in the network
- $Prim_G$ – image area covered by the (modified) goal)

$Quality_G = Certainty_G * Prim_G + RemSegm_G$, where

$Certainty_G$ =Certainty(Instance(predecessor goal)) * Priority(context-of–, spec–link)

$Prim_G = \sum_{Primitive \in PRIM(V)}$ (Area(Primitive) * Priority(Primitive)) $/Segm$
$Segm = \sum_{Primitive \in SEGM}$ Area(Primitive) * Priority(Primitive)

$RemSegm_G = Segm - Data_G$

For each image primitive a priority value is given, which reflects the certainty and selectivity of detected segment.
$Priority_G$ is equal to the difference between the longest path in the goal concept space from an initial to terminal goal concept and the path of actual goal.

Judgement vector of level (2)–node N:

- $Quality_{INST}$ – quality of the INST(N) set
- $Certainty_{INST}$ – certainty of the INST(N) set
- $Priority_P$ – priority of the realized instantiation path for GOAL1
- $Prim_{INST}$ – image area covered by the INST(N) set

The quality of the INST set should be monoton. But if new instances are added to INST, the $Certainty_{INST}$ may be decreased, whereas the $Prim_{INST}$ has a not decreasing value.
$Quality_{INST} = Certainty_{INST} * Prim_{INST} + RemData_{INST}$ where

$Certainty_{INST} = MIN_{I \in INST(N)}$ Certainty(I)

The certainty of a single instance is a function of three components:
a) syntactical component – certainties of part-instances (or the probability of a primitive)
b) contextual component – the priorities of part- or concrete-concepts (expressing the uniqality of a concept in the Model)
c) geometrical component – the degree to which numerical features of a concept and its structural relation are satisfied by the concrete- and part-instances

If all the above components are valued to the intervall $< 0, 1 >$, then the Certainty(I) is a monotonically decreasing function:
Certainty(I) = $MIN_{i \in \{Concretes(I) \cup Parts(I)\}}$ (Certainty(i) * Priority(i)) * Middle-Disparity(I)

The competitive instantiation paths for a goal concept are judged by means of *priority*. It is defined as the minimum of priorities of modality sets, that constitute the actual instantiation path for current goal.
$Prim_{INST}$ is computed for all primitives covered by the partial description INST, similar to the computation of $Prim_G$ for space (1).

Judgement vector of level (2)–node O:

- $Quality_{CI}$ – quality of the complete instance of GOAL1(O)
- $Certainty_{CI}$ – certainty of the complete instance of GOAL1(O)
- $Priority_{C}I$ – priority of the realized optional instantiation path for the optional links of extended GOAL1(O)
- $Covering_{INST}$ – combination of priorities of modality sets on the optional path with the certainty of instances generated on the space (3)–level

Chapter 7

Conclusion

There is a need of uniformity in symbolic vision processing, that is caused by large knowledge domains and by complexity of processing. The model retrieval operations are complex in vision, because most frequently subgraphs or other complex patterns are matched. We need to search for many hypotheses in parallel also. An uniform representation and inference techniques for all symbolic processing problems such as analysis, model aqusition and model rendering can greatly reduce the investments needed.

In this report it was shown, how a knowledge–based system can be used for solving the object recognition problem in computer vision. The single–image description of range images as input data was considered. The use of a semantic network–model has following features:
– An object representation compatible to the CSG–defined solids is provided
– Two complementary processes – bottom–up analysis and top–down synthesis (model rendering) are integrated and expressed in an uniform way
– The knowledge is utilized by domain–independent rules for instantiation and modification
– An optimal search in the space of competitive partial scene descriptions is the base of the recognition strategy

Bibliography

[1] Ballard D.H., Brown C.M.: **Computer Vision.** Prentice Hall, Englewood Cliffs, N.J., 1982.

[2] Besl P.J., Jain R.C.: *Three–Dimensional Object Recognition.* **ACM Computing Surveys, 17**(1985), 74-145.

[3] Shirai Y.: **Three–Dimensional Computer Vision.** Springer Series - Symbolic Computation, Springer Vg., Berlin, 1987.

[4] Rosenfeld A.(Ed.): **Machine Intelligence and Pattern Recognition, 3: Techniques for 3–D machine perception,** North-Holland, Amsterdam, 1986.

[5] Nilsson N.J.: **Principles of Artificial Intelligence.** Springer Vg., Berlin, 1982.

[6] Niemann H.: *Wissensbasierte Bildanalyse.* **Informatik–Spektrum,** 8(1985), 201-214.

[7] Niemann H., Bunke H.: **Künstliche Intelligenz in Bild– und Sprachanalyse.** Teubner Vg., Stuttgart, 1987.

[8] Sagerer G., Kummert F.: *Knowledge Based Systems for Speech Understanding.* In:[Niemann H. et al (Ed.): **Recent Advances in Speech Understanding and Dialog Systems.** NATO ASI Series, Vol. F46, Springer Vg., Berlin 1988], 421–458.

[9] Niemann H., Sagerer G., Schröder S., Kummert F. : *ERNEST: A Semantic Network System for Pattern Understanding,* IEEE Trans PAMI, 1990 (submitted for publication).

[10] Radig B.: *Image sequence analysis using relational structures.* **Pattern Recognition, 17**(1984), No.1, 161–167.

[11] Niemann H.: *Hierarchical graphs in patterns analysis.* **Proc. 5th ICPR** (Miami Beach, Fl. USA, 1980), 213–216.

[12] Barrow H.G., Ambler A.P., Burstall R.M.: *Some techniques for recognising structures in pictures.* In: [Watanabe P.(Ed.): **Frontiers of Pattern Recognition,** Academic Press, New York, 1972], 1–29.

[13] Lu S., Wong A.K.C.: *Analysis of 3–D scenes with partially occluded objects for robot vision.* Proc. 9th ICPR (Rome, 14-17 Nov 1988), IEEE 88CH2614–6, 303–308.

[14] Kasprzak W.: *A Linguistic Approach to 3–D Object Recognition,* **Computer & Graphics, 11**(1987), 427-443.

[15] Minsky M.: *A framework for representing knowledge.* In:[Winston P.H.(Ed.): **The Psychology of Computer Vision.** McGraw Hill, New York, 1975], 211–277.

[16] Quillian R.M.: *Semantic Memory.* In:[Minsky M.(Ed.):**Semantic Information Processing,** The MIT Press, Cambridge, MA, 1968], 216–270.

[17] Fahlman S.E.: **NETL: A System for Representing and Using Real–World Knowledge.** The MIT Press, Cambridge, MA, 1979.

[18] Brooks R.A.: *Symbolic Reasoning among 3–D Models and 2–D Images.***Artificial Intelligence, 17**(1981), 185-348.

[19] Hanson A.R., Riseman E.M.: *VISIONS, A Computer System for Interpreting Scenes.* [In: Hanson A.R., Riseman E.M. (Eds.) **Computer Vision Systems.** Academic Press, New York, 1978], 303-333.

[20] Riseman E.M., Hanson A.R.: *The VISIONS Image Understanding System.* [In: Brown Ch. (ed.): **Advances in Computer Vision.** Lawrence Erlbaum Ass. Pub., Hillsdale, N.J. 1988], 6-103.

[21] Herman M.: *Representation and Incremental Construction of a Three–Dimensional Scene Model.* In: [4], 149-183.

[22] Lee H–C., Fu K–S.: *Generating object descriptions for model retrieval.* **IEEE Trans Patt Anal Mach Intell, PAMI-5**(1983), 5(Sept.), 462-471.

[23] Hwang V.S.-S., Davis L.S., Matsuyama T.: *Hypothesis integration in image understanding systems.* **Computer Vision, Graphics and Image Processing, 36**(1986), 321-371.

[24] Kuan D.T., Drazovich R.J.: *Model-Based Interpretation of 3–D Range Data.* In: [4], 219-230.

[25] Yokoya N., Levine M.D.: *A Hybrid Approach to Range Image Segmentation.* **Proc. 9th Int. Conf. on Pattern Recognition** (Rome, Italy, 17-20 Nov 1988), 1988, 1-5.

[26] Hoffman R., Jain A.K.: *Segmentation and Classification of Range Images.* **IEEE Trans Patt Anal Mach Intell, PAMI-9**(1987), No.5, 608-620.

[27] Duda R.O., Nitzan D., Barrett P.: *Use of range and reflectance data to find planar surface regions,* **IEEE Trans Patt Anal Mach Intell, PAMI-1**(1979), 3(July), 254-271.

[28] Milgrim D.L., Bjorklund C.M.: *Range image processing and planar surface extraction.* In: [**Proc. 5th Int. Conf. on Pattern Recognition** (Miami, Fla.,Dec.1-4), IEEE, New York, 1980], 912-919.

[29] Henderson T.C., Bhanu B.: *Three point seed method for the extraction of planar faces from range data.* **Proc. Workshop on Industrial Applications of Machine Vision** (Research Triangle Park, N.C., May), IEEE, New York, 1982, 181-186.

[30] Hebert M., Ponce J.: *A new method for segmenting 3–D scenes into primitives.* **Proc. 6th Int. Conf. on Pattern Recognition,** (Munich, West Germany, Oct. 19-22), IAPR and IEEE, New York, 1982, 836-838.

[31] Besl P.J., Jain R.C.: *Invariant surface characteristics for 3D object recognition in range images.* **Computer Vision, Graphics and Image Processing, 33**(1986), No. 1, 33-80.

[32] Ozaki Y., Sato K., Inokuchi S.: *Rule–Driven Processing and Recognition from Range Image.* **Proc. 9th Int. Conf. on Pattern Recognition** (Rome,Italy,17-20 Nov), 1988, 804-807.

[33] Sugihara K.: *Range–data analysis guided by junction dictionary.* **Artificial Intelligence, 12**(1979), 41-69.

[34] Inokuchi S., Nita T., Matsuday F.,Sakurai Y.: *A three-dimensional edge–region operator for range pictures.* **Proc. 6th Int. Conf. on Pattern Recognition** (Munich, West Germany, Oct. 19-22), IAPR and IEEE, New York, 1982, 918-920.

[35] Mitiche A., Aggarwal J.K.: *Detection of edges using range information.* **IEEE Trans Patt Anal Mach Intell, PAMI-5**(1983), 2(Mar.), 174-178.

[36] Rao K., Nevatia R.: *Computing Volume Descriptions from xparse 3–D Data.* **Int J. of Computer Vision,** 2(1988), 33-50.

[37] Grimson W.E.L., Lozano–Perez T.: *Model–Based Recognition and Localization from Sparse Three–Dimensional Data.* In: [4], 113-148.

[38] Grimson W.E.L., Lozano–Perez T.: *Localizing Overlapping Parts by Searching the Interpretation Tree,* **IEEE Trans Patt Anal Mach Intell, PAMI-9**(1987), No.4, 469-482.

[39] Bhanu B.: *Representation and shape matching of 3–D objects.* **IEEE Trans. Pattern Anal Machine Intell, PAMI-6**(1984), 3(May), 340-350.

[40] Faugeras O.D., Hebert M.: *The Representation, Recognition and Positioning of 3–D Shapes from Range–Data.* In: [4], 13-51.

[41] Oshima M., Shirai Y.: *Object Recognition using Three–Dimensional Information.* **IEEE Trans Pattern Anal Machine Intell, PAMI–5**(1983), 353-361.

[42] Dhome M., Kasvand T.: *Polyhedra recognition by hypothesis acumulation.* **IEEE Trans Patt Anal Mach Intell, PAMI–9**(1987), No.3, 429-438.

[43] Bolles R.C., Horaud P.: *3DPO: A three–dimensional part orientation system.* **Int J. of Robotics Research,** 5(1986), 3, 3-26.

[44] Umeyama S., Kasvand T., Hospital M.: *Recognition and Positioning of Three–Dimensional Objects by Combining Matchings of Primitive Local Patterns.* **Computer Vision, Graphics and Image Processing,** 44(1988), 58-76.

[45] Bhanu B., Chen C.C.: *CAD based 3D Object Representation for Robot Vision* . **IEEE Computer Magazine,** 20 (1987), No.8, 19–35.

[46] Lin W.-C., Chen T.-W.: *CSG–Based Object Recognition using Range Images* . **Proceedings of the 8-th ICPR,** 1988, IEEE, 99-103.

[47] Niemann H. et al.: *A Knowledge Based System for Analysis of Gated Blood Pool Studies.* **IEEE Trans Patt Anal Mach Intell, PAMI-7** (1985), 245-259.

[48] Niemann H., Sagerer G., Eichhorn W.: *Control strategies in a hierarchical knowledge structure.* **Int J. of Pattern Recognition and Artificial Intelligence,** 2(1988), No.3, 557-572.

[49] Sagerer G.: . Habilitation Paper, 1989.

List of figure captions

List of table captions