Chapter 5
Model-Based 3D Object Recognition in RGB-D Images

Maciej Stefańczyk and Włodzimierz Kasprzak

Abstract A computational framework for 3D object recognition in RGB-D images is presented. The focus is on computer vision applications in indoor autonomous robotics, where objects need to be recognized either for the purpose of being grasped and manipulated by the robot, or where the entire scene must be recognized to allow high-level cognitive tasks to be performed. The framework integrates solutions for generic (i.e. type-based) object representation (e.g. semantic networks), trainable transformations between abstraction levels (e.g. by neural networks), reasoning under uncertain and partial data (e.g. Dynamic Bayesian Networks, Fuzzy Logic), optimized model-to-data matching (e.g. constraint optimization problems) and efficient search strategies (switching between data- and model-driven inference steps). The computational implementation of the object model and the object recognition strategy is presented in more details. Testing scenarios deal with the recognition of cups and bottles or household furniture. Conducted experiments and the chosen applications confirmed, that this approach is valid and may easily be adapted to multiple scenarios.

5.1 Introduction

With the newly available sensors that generate RGB-D images (3D point clouds and corresponding color images) of already reasonable quality, 3D image analysis methods are intensively being developed [1, 2]. A low-level processing of such data is usually a model-independent one and it leads to the creation of 3D maps of the environment (typically voxel- or surfel maps) [3–5]. In turn, the Ontology level of an agent system considered in AI operates on high-level symbolic entities like complex

M. Stefańczyk (✉) · W. Kasprzak
Institute of Control and Computation Engineering, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
e-mail: M.Stefanczyk@elka.pw.edu.pl

W. Kasprzak
e-mail: W.Kasprzak@elka.pw.edu.pl

H. Kwaśnicka and L. C. Jain (eds.), *Bridging the Semantic Gap in Image and Video Analysis*, Intelligent Systems Reference Library 145,
https://doi.org/10.1007/978-3-319-73891-8_5

objects and actions. There is a need for a methodology and implementation for mid-level symbolic processing of 3D images that reliably closes the gap between these two representations.

The *knowledge-based* paradigm has been intensively studied for in the past, but preferably for 2-D image analysis (e.g. [6–8]) and has not yet been really considered for processing of RGB-D data.

Recently developed *Deep Neural Networks* (DNN) and *deep learning* techniques, are mostly successful for appearance-based object classification. They approximate functions, which apparently transform sensor data into numeric features either into segments or directly into object instances or classes [9, 10]. This is of importance when complex algorithms or functions need to be defined and implemented. Although the DNNs were applied to find bottom-up image transformations, the research on modelling of context information and top-down constraints in DNNs has also begun [11, 12]. Especially when 3D objects need to be recognized in a multi-object environment, it is crucial to explore physical and contextual object relations, like occlusion relations and the probability of common appearance in given environment. Graphical and stochastic models have proved suitable to handle such cases [13, 14]. It is still an open question whether neural networks techniques can deal with symbolic object-level and ontology-level concepts in order to mimic logical reasoning processes. Here, the knowledge-based approach leads straightforward to adequate solutions [15].

In particular, our focus is on basic scenarios for 3D object recognition that are explored in service and social robotics [16, 17]: human pose recognition, obstacle recognition/avoidance and grasping/manipulating of objects (Fig. 5.1).

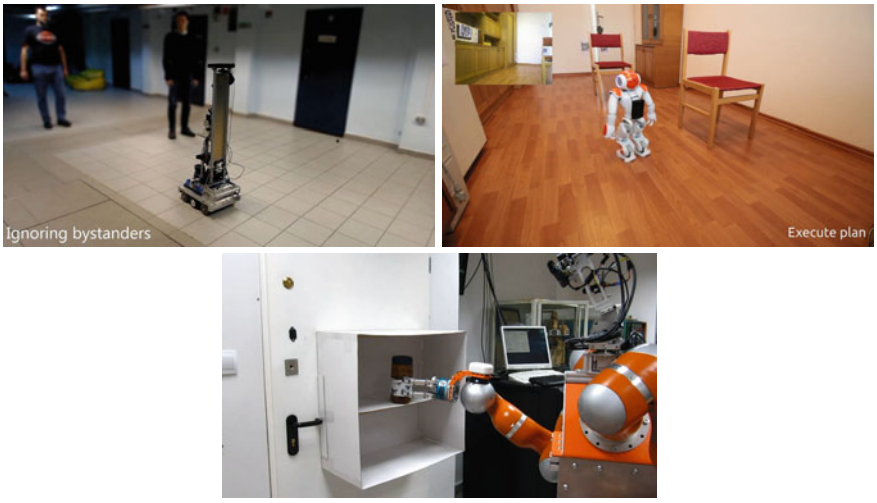

Fig. 5.1 Basic scenarios for 3D object recognition in service and social robotics: human recognition, obstacle avoidance and object grasping

So far, 3D object recognition in RGB-D images follows a data-driven strategy and mainly identifies a particular known object. Some software packages, preferably available in the ROS (Robot Operating System) programming environment, are listed next. MOPED [18], a real-time Object Recognition and Pose Estimation system, recognizes objects by comparing point-based features (e.g. SIFT, SURF) and their geometric relationships with rigid 3D object models (defined by point clouds). LINEMOD [19] ("multi-modal templates for texture-less object detection") is detecting texture-less 3D objects located in a strongly textured background. Textured Object Detector—is based on the standard "bag of features" technique [20]. During training, in images containing different views of the object, image features are extracted and their descriptors are obtained. For each of those features, the 3D position is also stored. Transparent Object Detector [21] is a pipeline that can detect and estimate poses of transparent objects, given a point cloud model of an object. The ODUFinder system [22] can detect and recognize textured objects in typical kitchen scenes. The models for perceiving the objects to be detected and recognized can be acquired autonomously using the robot's camera as well as by loading large object catalogs into the system. Richtsfeld et al. [2] developed an effective object model learning approach based on surface grouping in RGB-D data. But still, object instances are modelled and not their generic types.

A variety of *model-based* techniques have been developed in order to recognize 3D objects from images—hierarchical models [23], among them deformable part-based models [24] and probabilistic graphical models [14] appear to be most successful. In our paper, a model-based approach is proposed that is related to the principles of above techniques.

First, we focus on the 3D object representation and modelling issue. A discussion of knowledge hierarchy levels in object recognition systems is provided in Sect. 5.2, while Sect. 5.3 deals with 3D modelling in RGB-D data. In Sect. 5.4, our framework for 3D object recognition is introduced. The system's concept and its main element—the knowledge representation techniques and inference rules—are presented here. System implementation is summarized in Sect. 5.5. The work is illustrated in Sect. 5.6 by an application of robot vision in a household environment.

5.2 Knowledge Representation Hierarchy

In this section, a review of some approaches to general 3D object representation in images is presented and our solution, suitable for RGB-D images, is given. General levels of information representation (also called *categories* of representation entities) for 3D object recognition in images are discussed.

5.2.1 Related Work

Early image analysis systems mainly used linear features and wire-frame models, so categorization was made accordingly. Marr [25] distinguished four main conceptual levels of information representation. The first one is the IMAGE—represented as an array of point intensities. Second is a PRIMAL SKETCH, which contains some basic structures extracted from the image, like edge segments, discontinuities of intensity, gradient zero crossings, etc. Based on these features a $2\frac{1}{2}$ SKETCH is created, describing the visible surfaces in terms of contours, orientation and roughly estimated depth (all expressed in the viewer coordinate system). The last level is the 3D MODEL, describing the shapes and their spatial organization in an object-centered coordinate system. An object is composed of both volumetric and surface primitives and is arranged hierarchically.

Lowe [26] proposed a slightly different categorization in his system, where instead of the $2\frac{1}{2}$ sketch 2D PERCEPTUAL GROUPINGS are used. This requires a clustering of image features, obtained in the previous step, into some consistent groups. This extension made the description more general, as virtually any feature can be used and not only linear segments like before. Lowe also added an explicit VERIFICATION STEP, connecting 3D models with low-level image features. This in turn put some restrictions on the features and model used—there must be defined a method for object back-projection onto the picture. The hypothesis-generation and -verification cycle as a basic 3D scene recognition strategy was modelled formally by Kasprzak [13] as a bi-directional syntactic-semantic derivation using an attributed structure grammar.

Data representation categories correspond to different processes transforming data from one form to another. Forsyth [27] distinguished EARLY VISION, consisting not only of basic operations like image preprocessing or edge detection, but also texture description and depth reconstruction from stereopsis or structure from motion. His MID-LEVEL VISION is responsible for clustering and segmentation, fitting objects to segments and tracking them. HIGH-LEVEL VISION is meant to be the place, where data is collected from multiple measurements. Hypotheses are generated and verified here. Object detection and recognition at this level is done using complex classifiers. Relationships between detected objects can also be described.

Gonzalez [28] made explicit definition of the processes by defining their interfaces—data types that are used on the input and output of a proces. For LOW-LEVEL PROCESSES a *picture* is used as both the input and output. Processes are elementary picture operations, like image filtering. MID-LEVEL OPERATIONS on pictures are responsible for their segmentation into consistent groups, their description reducing representation dimensionality and also the classification (or recognition) of those segment groups into individual objects. The output of this process, usually taking the form of classified objects characterized further by vectors of numbers (attributes), is supplied to the HIGH-LEVEL VISION—a symbolic processing level that is responsible for image understanding and performs cognitive reasoning.

5.2.2 Proposed RGB-D Data Hierarchy

In case of depth data, sometimes additional processing steps are required, which are somewhere in between preprocessing and feature extraction. Examples of these operations are:

- normal vector calculation or curvature estimation for a surface patch,
- conversion from a depth map to full XYZ coordinates of a point cloud,
- transformation between different coordinate systems.

As a result, it seems reasonable to put an additional DATA EXTENSION layer between the signal- and feature extraction layers. Wrapping up, data representation in our 3D object recognition system is composed as a hierarchy of 6 layers, given below.

Hardware layer. It contains actual devices for data acquisition (cameras, sensors).

Signal layer. It is responsible for image pre-processing and data preparation for feature extraction (e.g. computing edge images or labeling consistent regions). These operations need no any external information and can be run using only one (current) picture.

Extension layer. It contains processes for computation of new data representation (from those returned by the sensor) or transformation of those using some external information (like sensor position or context images). These are operations like background subtraction, normal vector calculation, depth extraction from stereo images, coordinates transformation etc. Processes from extension and signal layer can be interleaved.

Feature extraction layer. It extracts condensed, numerical information from pictures, such as feature points, edges or blob segments. The produced information may vary from simple parameters, like line end points or segment mask, through some statistical information, like mean color or surface convexity [29], to higher-level interpretation, like parameters of inscribed surfaces [30].

Object recognition layer. It gathers segments and features computed by the lower layer and composes them to form an object of interest, based on some kind of provided model (at this stage the recognition is limited to single, isolated, objects). Object recognition processes can influence the way lower level processes work (e.g. changing parameter settings of feature extraction functions in order to return more or less crisp data).

Cognitive layer. Here a higher-level (symbolic) reasoning about the scene occurs (e.g. physical and contextual relations between objects are explored). This layer is also responsible for accumulating information in time (e.g. to allow lateral processing that improves the estimation of object parameters from multiple measurements).

Particular implementations of the processes located in low- and mid-level layers depend on the chosen form (*modality*) of the object model. Such different modalities (e.g. 2D edge model, 3D surface model) require different operations on previous

layers. On the other hand, an object recognition system should be generic and should allow usage of generic model description, making it possible to recognize different instances of the same kind of objects (like different sizes of jars or widths of doors).

5.3 3D Object Modelling

In this section, possible 3D object modeling modalities are discussed. The focus is on computer vision applications in indoor autonomous robotics, where objects need to be recognized either for the purpose of being grasped and manipulated by the robot, or where the entire scene must be recognized to allow high-level cognitive tasks to be performed.

Both steps require different modalities of 3D object models: geometric modelling (of physical shapes) or conceptual modelling (aggregations of parts).

5.3.1 Geometric Primitives

Grasp planners [31] work using geometric models of actual instance of the object, and this must be in a form compatible with physical engines used in a machine process, like triangular meshes or, even better, a composition of basic shapes. Methods for representation of geometric primitives can be generally divided into two main groups—DISCRETE and CONTINUOUS.

Discrete description keeps information about some finite number of elements or features, sampled from the original solid/object.

- When points are sampled from the surface of the object a POINTCLOUD is produced. The data structure of a point consists of its spatial coordinates (usually given in the Cartesian coordinate system), but it also can include other information, like the surface color or normal vector of a surface patch around this point. Points can be further expanded to surface elements (called surfels) [5], which are small surface patches approximated by discs.
- When information about volume of the object is crucial, another representation can be used, utilizing some volumetric shapes instead of points. Those elements, called VOXELS, are usually modeled with cubes, formed in either regular grid (with every element having the same size) or hierarchic structure (like octree), allowing for better approximation of complicated shapes with smaller number of elements.

Main advantage of discrete representations is the ease of model creation—in vast majority of cases depth sensors return information in form easily convertible to pointclouds. Hence, models can be built from few object views only [32]. The discrete model accuracy is proportional to the density of the pointcloud or a voxel grid, which is proportional to model size (verbosity). It must be noticed, that after the

image sampling step some information about the scene is lost (the surface between sampled points).

In contrary, continuous models represent the scene information by parametric functions of continuous spatial variables, allowing recovery of any point on the object's surface. One example of such model, described in [33], is a FUNCTIONAL model, where a shape is given by an equation specifying a continuous set of points.

- PARAMETRIC equations, in form of $F : T^m \Rightarrow X^n$ explicitly define the object's points in a n-dimensional space based on m free parameters. For three-dimensional objects $n = 3$, while for $m = 1$ curves are defined and for $m = 2$—surfaces. This representation form allows to model a broad range of shapes, from simple volumes to superquadrics. The ability to directly enumerate surface points makes it easy to convert such a model to a pointcloud with theoretically unlimited precision (density).
- In contrast, if the function is given in IMPLICIT form, $F(X^n) \Rightarrow Z$, it can be treated as a characteristic function for the described shape. Surface of the object is made of all points satisfying $F(X^n) = 0$, which is sometimes hard to calculate. On the other hand, this representation makes it very easy to check, whether point lays inside the given volume by checking the condition: $F(X^n) < 0$. Hence, it can easily be converted to a voxel grid. It also allows for a simple collision detection. Calculating objects intersection is also much easier using few implicit functions and checking them one by one.

The acquisition of functional models is computationally more expensive than of discrete models, as it requires some sort of surface fitting procedure, along with constrained set of shapes [34].

Another kind of continuous models are COMBINATORIAL models, locating itself between simple and complex object representations.

- Models can be created by combining a finite set of functional submodels using set operations. Using implicit functions in a way mentioned earlier, a MEMBERSHIP object definition can be created—for example, in terms of an intersection or sum of few functions. This way more complicated shapes can be created from simpler ones, without the need to use complex equations.
- TOPOLOGICAL representation is another kind from this family. Topological structure holds the spatial relations between subelements, like two faces touching each other or two points being connected with a line. Although the elements may be of any type (e.g. parametric surfaces with values from some bounded set), the most common type of this representation is a mesh, being closely related to discrete representations. Points are connected with edges, from which planar polygons (triangles are most common) are composed. Polygons are grouped into surfaces, and those into a final shape.

5.3.2 Complex Objects

Contrary to an object grasping and manipulation application, vision systems employed for cognitive tasks require models to be composed of parts that correspond (directly or via its parts) to segments that are possible to be detected in images (e.g. feature points or textures from color images, surface patches from range data). The hierarchic nature of the model is an added value, making it possible to build complex objects out of simpler ones.

- CONSTRUCTIVE SOLID GEOMETRY (CSG) is a way to describe objects using logical operations on primitives (like sum, difference, common part) and some geometrical modifiers (expansion, morphing). Representation of the primitives must be appropriate for this task, and functional models suits here the best.
- Another method is just a simple HIERARCHICAL model, where primitives are connected with each other using joints, and the transformation between them are done using homogeneous matrices. That kind of representation makes it possible to create objects with internal degrees of freedom (like cabinet with doors), as homogeneous transformations between parts can be parameterized. Similar strategy can be used to describe whole scene as one tree with multiple smaller object trees connected [35].

Another important factor, when talking about complex objects, is the ability to define different LEVELS OF DETAIL. When observing an object from far distance only the biggest parts are visible, and those can be also simplified, as finer details may disappear. The parts of the model can be differentiated—for coarse model only some global percepts can be used for description (like histograms of colors or silhouettes), whilst closer view can incorporate local texture descriptors and good quality depth data. When the distinction is made for only two levels—coarse and fine—the first one can be used for fast generation of object hypotheses, while the second level is used for hypothesis verification.

5.4 System Framework

5.4.1 Solution Principles

Object recognition is considered to be an intermediate image analysis level, located between the low-level image segmentation processes and the ontology level of a scene understanding process.

Both main computational paradigms, the knowledge-based one (e.g. model-based) and the neural network one (e.g. appearance-based), try to overcome the limitations of available 3D computer vision systems by concentrating on three basic design principles:

1. hierarchical framework-like architecture with increasingly abstract representation levels;
2. iterative control of object recognition by integrating bottom-up, top-down and lateral processing;
3. adaptability of the general framework to particular application domain by learning the object model and recognition strategy.

Following about design principles, in this section a 3D object recognition framework is developed, which integrates several methodologies, like proposed by us earlier [23, 36]: a generic (i.e. type-based) object representation (using semantic networks), trainable transformations between abstraction levels (performed by neural networks and deep learning techniques), techniques for reasoning under uncertain and partial data (e.g. Bayesian networks and Dynamic Bayesian Networks, Fuzzy Logic), an optimized model-to-data matching (e.g. constraint satisfaction and optimization problems) and efficient search strategies (controlling alternative realizations of data-driven hypothesis generation and model-driven hypothesis verification steps).

5.4.2 *Knowledge-Based Framework*

Knowledge-based systems are decomposed into two main parts: the *knowledge base* and the *control* [15]. Our particular system structure is depicted in (Fig. 5.2).

The knowledge base contains three elements: the *MODEL*, the *DATA* and *inference RULES*. In this approach, the model has a hybrid form, built around the structure and inference mechanism of a *semantic network*. Besides the declarative model and

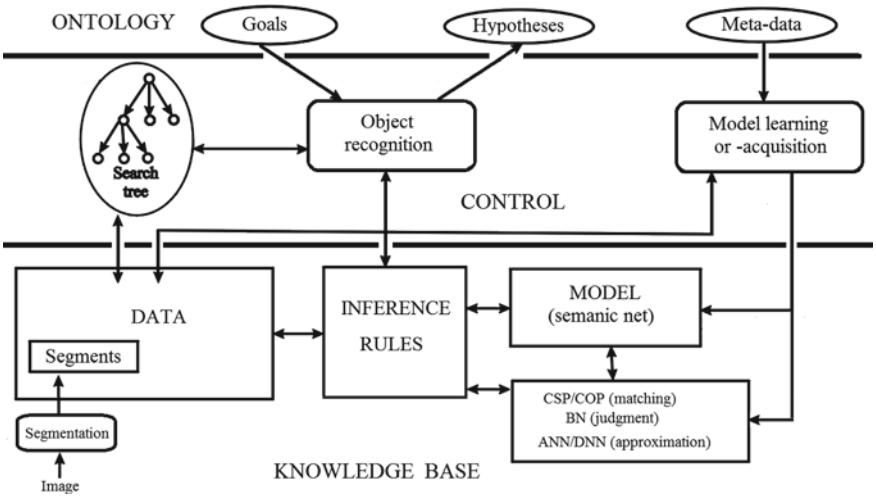

Fig. 5.2 Our knowledge-based framework

data structure expressed by the *concepts* and their interconnections in the semantic network, other techniques are here integrated: a dedicated *constraints satisfaction problem* for model-to-data matching, a *Bayesian network* for quality judgement of an *instance* or *constrained concept* and a *neural networks* (eventually *deep learning neural networks* for attribute computation.

The inference rules take the form of: "IF (condition) THEN add instance or constrained concept to DATA".

The DATA holds current symbolic descriptions of the signal (image) in form of instances and constrained concepts, generated initially by low-level image analysis (basically—image segmentation) and later as a result of the model-based inference process.

The CONTROL part performs a search in the space of competitive hypotheses, guided by their judgement values. In every step an available subset of data has to be matched with some model concept in order to satisfy the condition of some inference rules. Hence a lot of alternative decisions have to be controlled.

The model-to-data matching is seen as a specific *constraint satisfaction problem* or *constraint optimization problem*, but for many concepts it needs to be satisfied only partially (assuming a partial match).

The judgement of concept instance is estimated by a stochastic inference in a Bayesian net that is linked to given concept.

A general-purpose control strategy is defined by a space search algorithm.

5.4.3 Semantic Net

Common to semantic networks is the explicit structuring of domain knowledge along two hierarchies: the decomposition (vertical) hierarchy and the specialization (horizontal) hierarchy of concepts.

Starting from the pixel level the vertical hierarchy expresses increasingly abstract representation levels ("part" or "concrete" links). Simple elements are combined into more complex one, being parts of objects and scenes. Specialization links ("spec") represent inheritance relations between elements at the same abstraction level.

Every node (called "concept") represents some object category and it contains a parameter vector (called "attributes"), where every parameter is evaluated by some *term*, and every concept defines a set of constraints, evaluated by *predicates*, among its parts and related concepts.

A procedural part is added to the semantic network that implements the semantics of terms and predicates. It consists of functions for attributes and relations for predicates. In fact, a semantic network is an object-oriented form of a specific predicate logic. If we allow concept attributes to hold default values then such a semantic network represents a non-monotonic logic.

The part- and spec-links have an appropriate representation in logic. The relation, "{set of parts} $-part->$ concept C", is equivalent to a formula built around the

implication symbol, in straight direction, ($C_{part1} \wedge C_{part2} \wedge ... \wedge C_{partN} \Rightarrow C$), and in the reverse direction, ($\forall_{I \in 1,...,N}(C \Rightarrow C_{partI})$).

Similarly, the dependence, "base concept $-spec- >$ inherited concept", is equivalent to a formula: $C_{inherited} \Rightarrow C_{base}$

5.4.4 Bayesian Net

A Bayesian net (BN) is a simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions: (1) a set of nodes, one per stochastic variable; (2) a directed, acyclic graph (link means "direct influence")—incoming links of given node represent a conditional distribution for this node given its parents, $P(X_i|Parents(X_i))$. In the simplest discrete case, conditional distribution is represented as a conditional probability table (CPT), giving the distribution over X_i for each combination of parent values.

An illustration of a Bayesian net is shown in (Fig. 5.3)—it represents variables related to a Rubik_cube concept. An intermediate level in the model represents visible faces. The lowest-level concepts represent 9 color squares, that define the texture of a face. There are also evidence nodes that represent constraints between faces (fA, fB) and constraints between squares (A, B, D).

The score of a partial solution (assignment in terms of CSP), in which some variables X_i have already been assigned to image segments l_k but not all of them, is obtained due to stochastic inference in Bayesian net. For example the computation of posterior probability of a "cube" instance (that is a *cause* in terms of BN) given its parts (that are *evidences* in BN). For example, if segments are assigned to X_0 and X_1 then one need to compute the probability: $P(cube|X_0 = l_1, X_1 = l_2)$.

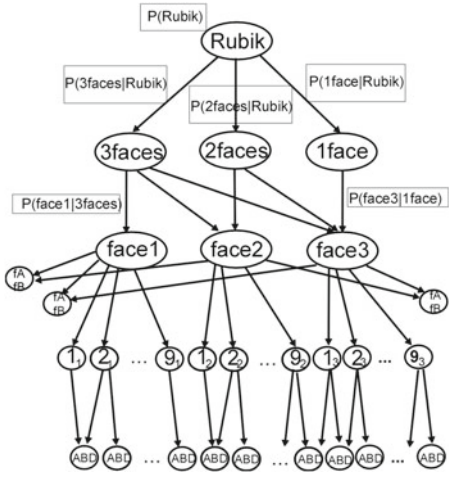

Fig. 5.3 A Bayesian net structure for concept: "Rubik_cube"

This leads to a summation of pdf over all domain values for remaining (non-evidence) variables, X_2, ..., X_l. Thus, scores of partial matches or a complete match, between image segments and model entities, are naturally obtained by the same evaluation method.

5.4.5 The Basic Control

The object recognition process is performed for given set of object concepts, called the GOALS G. This set can contain "concepts", "constrained concepts" or even "instances". Let M denotes concepts stored in the model base, while D is the current data set. In every single step, the basic control algorithm activates one of the five available inference rules, RULE_1, ... , RULE_5, for selected model concepts and data instances.

1. IF $G \neq \emptyset$ THEN perform a top-down goal concepts expansion (propagation of constraints), using inference RULE 4; ELSE perform a bottom-up hypothesis generation for concepts in M, based on important image segments in D, using RULE 5.
2. A bottom-up generation of "partial instances", that match the existing "constrained concepts" for obligatory parts of some modality of the selected model concept with the data instances (using RULE 1): $I_p(k) \leftarrow \{(part_i \in M_k; I_i \in D) | i \in oblig(M_k)\}$; where attributes of every instance $I_p(i)$ are $a = (S_k, R_k, t_k)$ (shape, rotation, translation);
3. Hypothesis verification: FOR every hypothesis $I_p(k)$ DO
   - constrain its remaining (non-obligatory) parts ("top-down" RULE 2) and match these parts with DATA: $I_e(k) \leftarrow \{(part_j \in M_k; I_j \in D) | j \in optional(M_k)\}$
   - Verify the hypothesis $I(k) \leftarrow (I_p(i) \bigcup I_e(k))$—create a "full instance" and re-compute its attributes a' (a "bottom-up" RULE 3).
4. Return the lattice of verified hypotheses, i.e. a graph where nodes represent hypotheses and arc—relations of mutual exclusion.

It depends on a particular search strategy (and current data and hypotheses) which step is selected and performed next.

5.5 System Implementation

The particular data types and predicates will be discussed that are implementations of nodes of the abstract semantic net (concepts) and the constraints between parts of a concept.

Two basic building blocks of the knowledge base, the MODEL and the DATA, are connected with two views of a 3D object. First is the "idealized" view, i.e. the object's type. The other one is the instance hypothesis, i.e. a set of parts (e.g. segments) recognized as an object of interest.

5.5.1 Model Structure

The *model M* is the "idealized" view of the object, describing its generic properties and allowing to recognize multiple realisations (instances) of this type of objects, like chairs of different sizes or different bottles, as long as they share some common features. A single object's model is the implementation of a dedicated *concept* from a semantic network. A model is built from PARTS P, constituting observable objects itself, CONSTRAINTS C, defining relations between those parts, ATTRIBUTES A, allowing a differentiation between instances and SCORE—a judgment of instance quality. Thus, a model is a tuple consisting of following entities:

$$M = \left\{ P = \{p_1 \dots p_n\}, C = \{c_1 \dots c_m\}, A = \{a_1 \dots a_k\}, score = \{s_1 \dots s_l\} \right\} \tag{5.1}$$

Parts have a pre-defined unique *role* in the model (like left leg or mug handle), while the constraints are expressed by relations between parts of a concept and are evaluated on attributes of these parts. Alternative "specialized" versions of a concept or alternative subsets of the parts of a concept (called as *modalities*), are illustrated by OR links on the diagram on Fig. 5.4.

The basic structure of a single model is represented by a graph on Fig. 5.5. Following sections describe every element of this diagram in details.

As a simple example, illustrating presented concepts, the mug object is used. Putting term *mug* into the image search engine yields a list different pictures (Fig. 5.6), but all of the presented objects possess some common elements. Every mug has a more or less toroidal handle and a main cylindrical part for liquid. They differ in size and color, but can be described using one generic model.

5.5.1.1 Parts

Part is some observable element of the object. It can be sometimes identified as a physical element, like the leg of a chair or a door knob, but in other cases it can be a more abstract thing, like the edge of a box or even a single point (or feature point) extracted from an object surface.

For each part p_i there must be assigned a CLASS ($class(p_i)$), i.e. another model representing this entities type. This defines, what kind of part it is, and whether it can be for example matched to a *cylinder* in applications, where one can observe 3D geometrical shapes, or matched with a *line* when edge-based processing is used, etc.

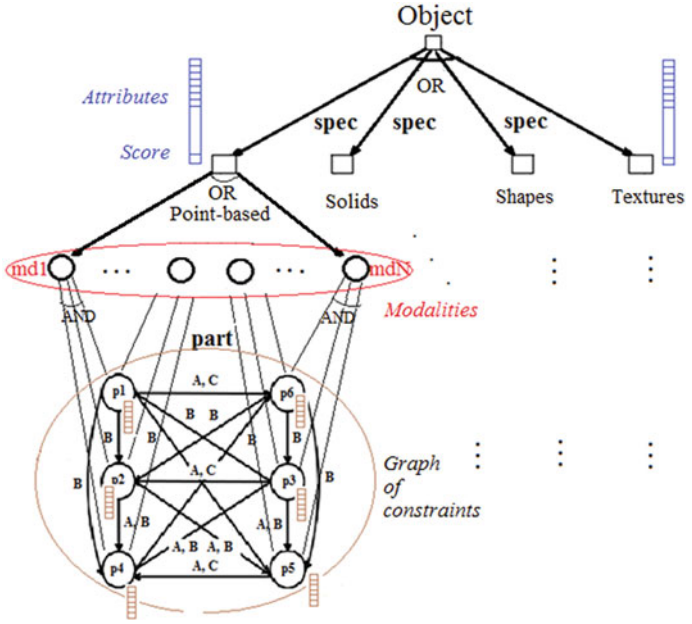

Fig. 5.4 Alternative 3D object models (specialized concepts of the "Object" concept) and typical structure of a single model

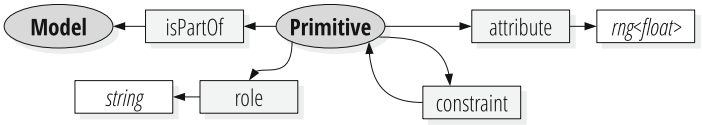

Fig. 5.5 The TBox class structure—abstract concept implementation

There can be, of course, many parts of the same class, like there are for example four legs in a chair.

To differentiate between parts of the model, each has to have defined a unique ROLE, which will be used as an identifier in further processing steps. The role of the part can be either a more abstract one, like a *left-edge* when describing geometrical shapes, or it can mimic part affordances, like a *handle* for the toroidal part and *body* for the cylindrical part of the mug model.

5.5.1.2 Attributes

Attributes describe properties of a model. In the model, an attribute is defined by its data type and the range of its allowed values. Only instances of such model with attribute values lying in given range can be considered as its proper instances. The

Fig. 5.6 Sample mugs retrieved by web image search

attribute calculation function itself can return any value from the domain D of the attribute's type:

$$Attribute(M) \in \mathscr{D} \tag{5.2}$$

The same attribute type can be used for multiple parts, but for each of them different range of possible values can be set. For example, color of the main part of the mug (modeled as a hue component) can be set to red, while the handle can be white. Another group of attributes are of geometrical nature. Typical mugs have radius in the range from 3 to 6 cm.

5.5.1.3 Constraints

In contrast to attributes, constraints are defined on some subset (at least with two elements) of parts, and they represent some relation between them. There could be logical constraints (like checking, if some parts have the same size), spatial ones (like checking, if two lines are parallel) or others.

Fuzzy set functions for constraint evaluation return values from the $[0 \ldots 1]$ range (instead of the Boolean values $\{True, False\}$), where 1 means full constraint satisfaction and 0 total inconsistency of given set of parts with examined relation. It enables to treat the result of constraint satisfaction check as an intermediate score in further processing steps, giving finally an overall score of the model's instance.

$$Predicate(p_1, \ldots, p_k) \in [0 \ldots 1] \tag{5.3}$$

For the simple mug model one can require that its handle intersects with the main part. The *intersects* constraint can be defined as a function returning 1 if the handle's (toroid) center lies on the surface of the main cylinder and gradually dropping to 0 when the toroid's center is farther than its radius from the main part. This function may be based on the distance between center of the toroid (with radius r) and the axis of the cylinder (with radius R). It looks like the one presented on Fig. 5.7. The final model structure for the mug is presented on Fig. 5.8.

5.5.2 Object Instances

A model of some object is a generic representation of its structure. When an observation is made, multiple segments can be extracted from it, and those can be classified as instances of some basic concepts (model entities), called as the *primitives* of symbolic representation. During the object recognition process, some of these primitives can be assigned to model parts (if their attributes are in desired ranges), and after satisfying the constraints of given model, they eventually lead to the creation of this model's instance (e.g. an object hypothesis).

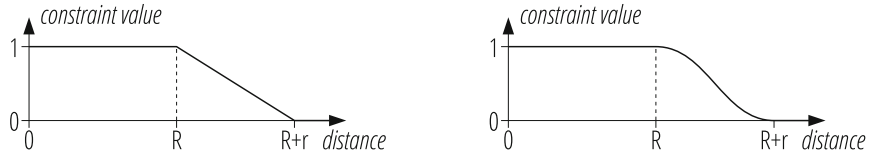

Fig. 5.7 Sample calculation functions for the *intersects* constraint

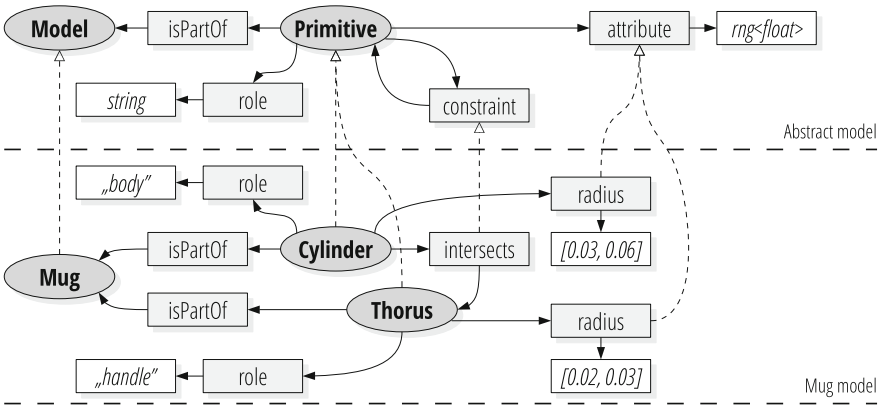

Fig. 5.8 A TBox representation of the "mug" model

5.5.2.1 Model-to-Data Matching

Each assignment of existing instances to model parts, made in a way that for each part of the model exactly one instance of proper class is selected, is called hypothesis $H(M)$. It can be defined as:

$$H(M) : \forall i \in \{1..n\} \exists j \in \{1..k\} : p_i \leftarrow inst_j \wedge class(p_i) = class(inst_j) \quad (5.4)$$

where n is the number of parts in the model and k is the number of already recognized instances ($inst_k$, e.g. segments). Overall hypothesis score is calculated by taking the product of all constraints for given assignment.

$$score\big(H(M)\big) = \prod_m eval(c_m, H) \quad (5.5)$$

The most naive way of generating hypotheses is to perform an *exhaustive search* in the entire space of possibilities, i.e. generating all variations of existing instances that match the model structure. This way, the number of generated hypotheses, that must be further checked for their scores, may be big or even huge:

$$|O| = \prod_{d \in D} |d| \quad (5.6)$$

where O is a set of object hypotheses and $D = \{d_i\}$ is a domain for particular part (i.e. set of all instances of the same class as given part). For each hypothesis its score is evaluated and the best ones are returned.

5.5.2.2 Matching by CSP/COP

To avoid full expanding of hypothesis before its verification (which is computationally very expensive), the matching problem can be treated as a *constraint optimization problem* (COP). Basic backtracking algorithm is used to build hypothesis step by step. After assigning a new variable (in our case assigning existing part-type instance from DATA to a yet unassigned model part) the hypothesis score is calculated and, if the score is lower than some existing threshold, current branch is pruned and the algorithm goes backward to search for other, better possibilities.

Classic CSP works until it finds the first solution satisfying all the constraints. As the score in our system can be anywhere between 0 and 1, we can compare two hypotheses and select the better one. CSP implementation is thus modified as follows. We keep track of N best hypotheses found so far (this list is empty when the search algorithm starts). At each step, the hypothesis score is calculated and, if it is lower than the worst from the list, this search tree branch is pruned. If a complete hypothesis is generated and its score is high enough, it is placed on the list. This way we achieve

the same effect as in exhaustive search, but with much better performance—a lot of hypotheses are rejected at early stage.

For efficiency reasons, this basic strategy is additionally supported by three techniques: selecting the most constrained variable first, selecting the highest-scoring value first, and making a forward check of the constraints.

5.6 Testing Scenarios

As an simple illustration, generic mug model will be used (similar to presented in previous sections), with two distinct modalities of incoming data: RGB image and RGB-D image. In first scenario, edge-based analysis [37] is applied, with two basic concepts—*Cylinder* and *Arc*, corresponding to two mug parts—*body* and *handle*. Second scenario uses depth data [38], and two basic surface concepts—*Cylinder* and *Thorus*.

5.6.1 Data Acquisition

As an input data, *Complex scene 3* from WUT Visual Perception Dataset [39] was used, which contains a recorded trajectory (77 points) "around the table". The set was acquired using Microsoft Kinect sensor, and it contains, for every recorded position, a pair of images, aligned with each other: RGB image (Fig. 5.9a) and depth map (Fig. 5.9b). The selected scene contains three cylindrical objects—two mugs and one coffee jar, as well as some other kitchen utensils.

The data was acquired with hand-held sensor, thus there is no ground-truth position data and the trajectory was recovered using visual odometry solution [5].

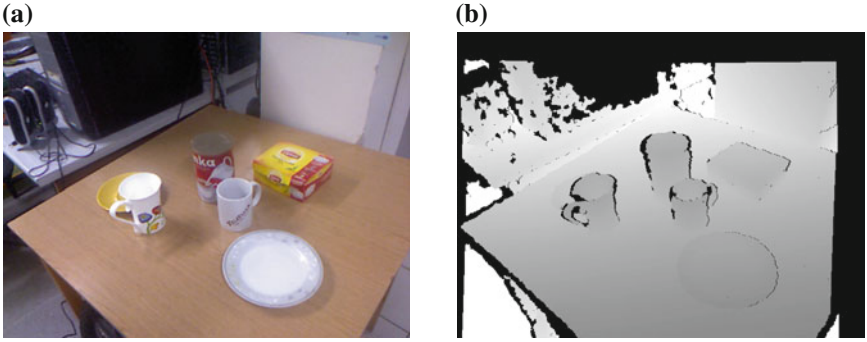

Fig. 5.9 Test scene: **a** RGB image, **b** depth map

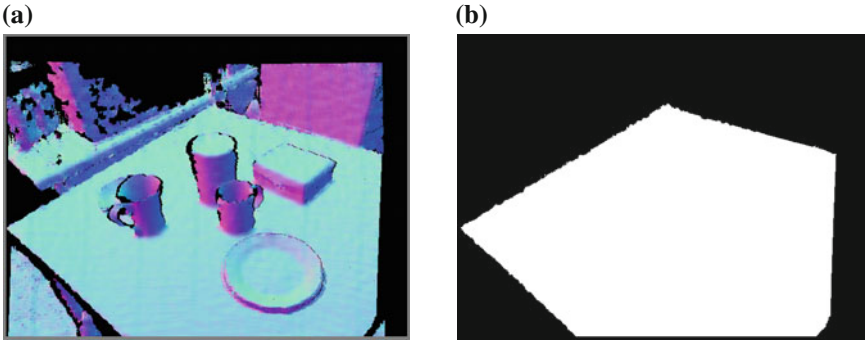

Fig. 5.10 Results of data extension: **a** calculated normal map; **b** mask of interesting scene part

5.6.2 Data Preprocessing and Extension

During preprocessing and extension phase, there are two steps worth mentioning. One is the calculation of surface normals. For every valid point in depth map, if it contains sufficiently big surrounding, it is used to calculate a vector perpendicular to the surface in given point (Fig. 5.10a). Another operation is mask generation. Based on information from control subsystem, search space can be restricted to a smaller area—in this case only objects on the brown table are interesting for us (Fig. 5.10b)

5.6.3 Segmentation

Edge-based analysis uses a two-step segmentation process [37]. At first, only linear segments are detected (arcs and lines), and then those are connected into more complex structures (cylinders in described scenario). To create those complex structures, the same hypothesis generation step is used, with a model describing cylinder appearance. Final segmentation result looks like shown on Fig. 5.11a.

Surface segmentation uses RanSaC to inscribe cylindrical and thoroidal surfaces into acquired 3D image (using points position in space as well as their normal vectors). Sample segmentation result for one view is shown on Fig. 5.11b.

5.6.4 Hypothesis Generation

Based on detected segments, initial mug hypotheses are generated for the example model containing two parts (*body* and *handle*) and one constraint between them (*near*). In edge based scenario, four hypotheses were generated (Fig. 5.12a). One proper hypothesis for the mug on the right-hand side, two competing correct hypothe-

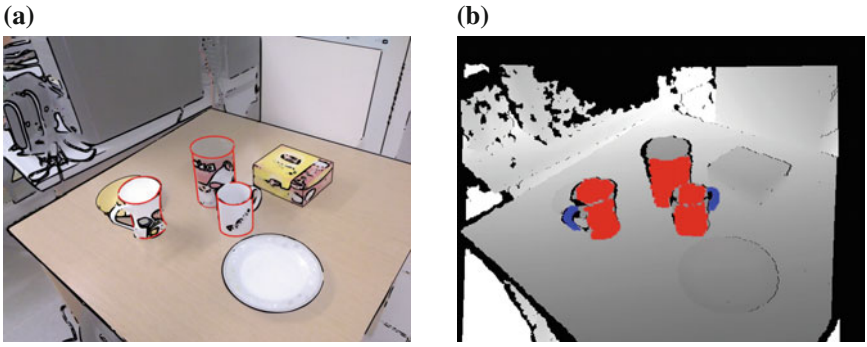

Fig. 5.11 Basic instances detected in segmentation step: **a** edges—cylinders (red) and arcs (black), **b** surfaces—cylinders (red) and thoroids (blue)

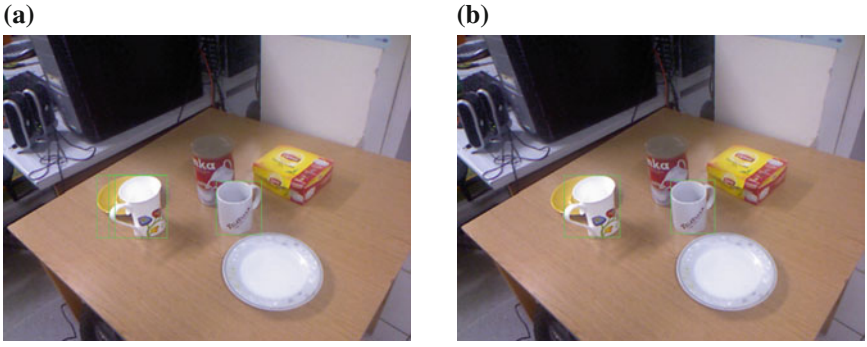

Fig. 5.12 Initial hypotheses: **a** edge-based, **b** surface-based

ses for the left-hand side mug (two different arcs for the *handle*) and one false hypothesis for the same mug (with plate taken as an *handle* part).

In second scenario, where surface-based analysis is used, only two hypotheses are generated, one for each mug on the scene. It is worth mentioning, that in both cases coffee-jar was not taken into account as candidate object because of lack of nearby *handle*-like segment.

5.6.5 Hypothesis Update and Verification

Initial hypotheses are tracked in consecutive images. When a new measurement comes in, apart from detecting new hypotheses, existing ones are matched against detected segments and their scores are recalculated. In edge-based scenario, this tracking step allows to detect false mug-plate hypothesis, as in subsequent views the underlying parts (cylinder and arc) move away from each other (Fig. 5.13a).

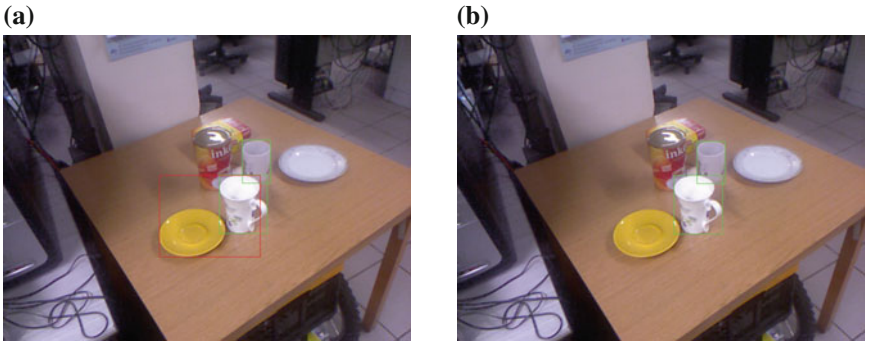

Fig. 5.13 Tracked hypotheses, green—good, red—bad: **a** edge-based, **b** surface-based

Second scenario was initialized with two proper hypotheses, so nothing is dropped in subsequent views. More measurements, however, are used to make better estimation of object parameters, for example position in space or size (using Kalman filter).

Detected (and updated) objects are returned to control subsystem for each incoming measurement.

5.6.6 Method Vulnerabilities

The proposed object recognition method is generic as it explores a generic object model. Its efficiency depends highly on the quality of particular domain model given as its input. It should be noted, that for poorly created models the recognition results may be disappointing, or the processing time can be very long. If the model is created with very high level of details and precision, e.g. every model element has highly limited range of accepted attribute values, even small sensor inaccuracies can lead to a rejection of all available image segments, resulting in a failure of object detection. On the other hand, when the model is defined with small number of constraints and very relaxed attribute restrictions, almost every image segment must be checked when building object hypotheses. This requires to expand many, if not all, possible assignments, which in turn leads to long processing times. Thus, it is important to select proper subsets of object features, small enough to keep a generic model, yet specific enough to be discriminative. Our current focus is on automated methods of creating such optimal models.

5.7 Conclusions

An application-independent generic model-based framework for object recognition in RGB-D images has been presented and verified on robot vision scenarios. It has clear advantages over existing, mostly data-driven and appearance-based approaches to object instance re-detection. First, it allows to identify what kind of knowledge is needed and to utilize existing meta-level knowledge (e.g. types of predicates and attributes commonly used for object description) to perform machine learning of model concepts (to learn concept types instead of memorizing individual instances). Secondly, common parts of object recognition systems can be pre-implemented, which increases the efficiency of system design and its implementation in different applications.

Another important advantage of proposed system is its human-oriented approach to object modelling. The decomposition of an object into simpler elements, named parts, makes it easier to further analyze the model. Using fuzzy constraint functions enables the user to focus on overall model creation, instead of performing a hand-made tuning of parameters.

Conducted experiments (described in Sect. 5.6) and the chosen applications (presented on Fig. 5.1) confirmed, that this approach is valid and may easily be adapted to multiple scenarios. In the article, we selected one example application of the system, simple enough to be easy to follow by the reader, yet covering two most popular data modalities—color images and depth measurements. Developed algorithms are independent on the data source, so exactly the same methods and algorithms can be used regardless of selected sensor. In fact, different data sources were tested (single cameras, stereo pairs, structured light), mounted in different spots (stand alone over the workbench, mounted on robots head or at the end of the arm, near the gripper).

Acknowledgements The manuscript preparation was supported by statutory funds of the Warsaw University of Technology.

References

1. Ren, X., Fox, D., Konolige, K.: Change their perception: RGB-D for 3-D modeling and recognition. IEEE Robot. Autom. Mag. **20**(4), 49–59 (2013)
2. Richtsfeld, A., Mörwald, T., Prankl, J., Zillich, M., Vincze, M.: Learning of perceptual grouping for object segmentation on RGB-D data. J. Vis. Commun. Image Represent. **25**(1), 64–73 (2014)
3. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: using kinect-style depth cameras for dense 3D modeling of indoor environments. Int. J. Rob. Res. **31**(5), 647–663 (2012). https://doi.org/10.1177/0278364911434148
4. Newcombe, R.A., Davison, A.J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., Fitzgibbon, A.: KinectFusion: real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 127–136. IEEE (2011)

5. Wilkowski, A., Kornuta, T., Stefańczyk, M., Kasprzak, W.: Efficient generation of 3D surfel maps using RGB-D sensors. Int. J. Appl. Math. Comput. Sci. (AMCS) **26**(1), 99–122 (2016). https://doi.org/10.1515/amcs-2016-0007
6. Hanson, A., Riseman, E.: The VISIONS image-understanding system. Adv. Comput. Vis. **1**, 1–114 (1988)
7. Hwang, V.S.S., Davis, L.S., Matsuyama, T.: Hypothesis integration in image understanding systems. Comput. Vis. Graph. Image Process. **36**(2–3), 321–371 (1986)
8. Niemann, H., Sagerer, G.F., Schroder, S., Kummert, F.: ERNEST: a semantic network system for pattern understanding. IEEE Trans. Pattern Anal. Mach. Intell. **12**(9), 883–905 (1990)
9. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1097–1105 (2012)
10. Socher, R., Huval, B., Bhat, B., Manning, C., Ng, A.: Convolutional-recursive deep learning for 3D object classification. Adv. Neural Inf. Process. Syst. **25**, 656–664 (2012)
11. Behnke, S.: Hierarchical Neural Networks for Image Interpretation. Lecture Notes in Computer Science, vol. 2766. Springer, Berlin (2003)
12. Lin, D., Fidler, S., Urtasun, R.: Holistic scene understanding for 3d object detection with rgbd cameras. In: IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1–8, 2013, pp. 1417–1424. IEEE Computer Society, ISBN 978-1-4799-2839-2 (2013)
13. Kasprzak, W.: A linguistic approach to 3-D object recognition. Comput. Graph. **11**(4), 427–443 (1987)
14. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge (2009)
15. Russel, S., Norvig, P.: Artificial Intelligence. A Modern Approach, 3rd edn. Prentice Hall, Upper Saddle River (2011)
16. Tsardoulias, E., Zieliński, C., Kasprzak, W., Reppou, S.: Merging robotics and AAL ontologies: the RAPP methodology. In: Progress in Automation, Robotics and Measuring Techniques. Advances in Intelligent Systems and Computing, vol. 351, pp. 285–297. Springer International Publishing (2015)
17. Zieliński, C., et al.: Variable structure robot control systems–the RAPP approach. Robot. Auton. Syst. **94**, 226–244 (2017). https://doi.org/10.1016/j.robot.2017.05.002
18. Collet, A., Martinez, M., Srinivasa, S.S.: The MOPED framework: object recognition and pose estimation for manipulation. Int. J. Robot. Res. **30**(10), 1284–1306 (2011)
19. Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., Lepetit, V.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 858–865. IEEE (2011)
20. O'Hara, S., Draper, B.A.: Introduction to the bag of features paradigm for image classification and retrieval (2011). arXiv preprint arXiv:1101.3354
21. Lysenkov, I., Rabaud, V.: Pose estimation of rigid transparent objects in transparent clutter. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 162–169. IEEE (2013)
22. Pangercic, D., Haltakov, V., Beetz, M.: Fast and robust object detection in household environments using vocabulary trees with sift descriptors. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World, San Francisco, CA, USA. Citeseer (2011)
23. Kasprzak, W., Kornuta, T., Zieliński, C.: A virtual receptor in a robot control framework. In: Szewczyk, R., Zieliński, C., Kaliczyńska, M. (eds.) Recent Advances in Automation, Robotics and Measuring Techniques. Advances in Intelligent Systems and Computing (AISC), vol. 267, pp. 399–408. Springer, Berlin (2014)
24. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)
25. Marr, D.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Henry Holt and Co., Inc., New York (1982)

26. Lowe, D.G.: Three-dimensional object recognition from single two-dimensional images. Artif. Intell. **31**(3), 355–395 (1987)
27. Forsyth, D.A., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall Professional Technical Reference (2002)
28. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice Hall, Upper Saddle River (2002)
29. Stefańczyk, M., Kasprzak, W.: Multimodal segmentation of dense depth maps and associated color information. In: Bolc, L., Tadeusiewicz, R., Chmielewski, L., Wojciechowski, K. (eds.) Proceedings of the International Conference on Computer Vision and Graphics. Lecture Notes in Computer Science, vol. 7594, pp. 626–632. Springer, Berlin (2012)
30. Richtsfeld, A., Morwald, T., Prankl, J., Zillich, M., Vincze, M.: Segmentation of unknown objects in indoor environments. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4791–4796. IEEE (2012)
31. Miller, A.T., Allen, P.K.: Graspit! a versatile simulator for robotic grasping. IEEE Robot. Autom. Mag. **11**(4), 110–122 (2004)
32. Łępicka, M., Kornuta, T., Stefańczyk, M.: Utilization of colour in ICP-based point cloud registration. In: Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015. Advances in Intelligent Systems and Computing, pp. 821–830. Springer, Berlin (2016)
33. Naylor, B.: Computational representations of geometry. In: Representations of Geometry for Computer Graphics, SIGGRAPH '94 Course Notes (1994)
34. Jaklic, A., Leonardis, A., Solina, F.: Segmentation and Recovery of Superquadrics, vol. 20. Springer Science & Business Media (2013)
35. Foote, T.: tf: the transform library. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6. IEEE (2013)
36. Kasprzak, W.: Integration of different computational models in a computer vision framework. In: 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 13–18 (2010). https://doi.org/10.1109/CISIM.2010.5643697
37. Stefańczyk, M., Pietruch, R.: Hypothesis generation in generic, model-based object recognition system. In: Szewczyk, R., Zieliński, C., Kaliczyńska, M. (eds.) Recent Advances in Automation, Robotics and Measuring Techniques. Advances in Intelligent Systems and Computing (AISC), vol. 440, pp. 717–727. Springer, Berlin (2016). https://doi.org/10.1007/978-3-319-29357-8_62
38. Wilkowski, A., Stefańczyk, M.: Detection and recognition of compound 3D models by hypothesis generation. In: Szewczyk, R., Zieliński, C., Kaliczyńska, M. (eds.) Recent Advances in Automation, Robotics and Measuring Techniques. Advances in Intelligent Systems and Computing (AISC), vol. 440, pp. 659–668. Springer, Berlin (2016). https://doi.org/10.1007/978-3-319-29357-8_57
39. Stefańczyk, M., Laszkowski, M., Kornuta, T.: WUT visual perception dataset-a dataset for registration and recognition of objects. In: Challenges in Automation, Robotics and Measurement Techniques. Advances in Intelligent Systems and Computing (AISC), vol. 440, pp. 635–645. Springer, Berlin (2016)