Adv Syst Sci Appl 2023; **02**:24–45
Published online at https://ijassa.ipu.ru.

Object Recognition by a Minimally Pre-Trained System in the Process of Studying the Environment

Dmitry Maximov*, Sekou Diane

Trapeznikov Institute of Control Science, Russian Academy of Sciences, Moscow, Russia

Abstract: We refine a method for describing and evaluating a previously proposed process of studying an abstract environment by a system (robot). In the process, we do not model any biological cognition mechanisms and consider the system as an agent (or a group of agents) equipped with an information processor. The robot (agent) makes a move in the environment, consumes information supplied by the environment, and gives out the next move (thus, the process is considered as a game). The robot moves in an unknown environment and should detect new objects located in it and recognize them. In this case, the system should build comprehensive images of visible things and memorize them if necessary (and it should also choose the current goal set). The main problems here are object recognition and the assessment of information reward in the game. Thus, the main novelty of the paper is a new method of evaluating the amount of visual information about the object as the reward. In such a system, we suggest using a minimally pre-trained neural network to be responsible for the recognition: at first, we train the network only for Biederman geons (geometrical primitives). Training sets of geons are generated programmatically and we demonstrate that such a trained network recognizes geons in real objects quite well. Sets of geons connected with objects (schemes) are used as the rewards. We also expect to generate procedurally new objects from geon schemes obtained from the environment in the future and to store them in a database.

Keywords: Robot intellect, Cognition, Goal Lattice, Conway game semantics, Synthesis of training sets, Image classification

1. INTRODUCTION

The Universal Artificial Intelligence (UAI) is a unifying framework and a general formal foundational theory for artificial intelligence investigations [10]. Its primary goal is to give a mathematical answer to the question: what is the right thing to do in an unknown environment, and how can an intelligent system learn to behave in the environment? Such a learning process should be active in this case.“With active learning, the [robotic] system may deviate from the implementation of the main application and perform actions aimed at collecting information about the environment.” [7].

Investigations in the field are focused on systems which *act* rationally. The artificial intelligence is represented in the approach as an information processor that consumes and gives out information. The theory also tries to answer, in general, the question, “how can a system composed of relatively unintelligent parts (say, neurons or transistors) behave intelligently?” [6].

A formal description of the most intelligent agent behaviour, in the sense of some intelligence measure, is suggested in the UAI framework [17], [18]. The framework specifies how an agent *interacts* with an environment. The model is based on probabilistic modelling of the environment, and determination of the next system move, based on previous experience.

*Corresponding author: dmmax@inbox.ru

Also, it is based on a numerical estimation of the system position reward, and the expected reward maximization along the trajectory. However, the method to obtain this numerical estimation is absent.

It has been demonstrated in [22], [23], [21], that the existence of a structure (a lattice structure or, else, a monoid structure) in the set of tasks of the system [22], [23] or its goals [21] is sufficient for the system to behave quite reasonably. The behaviour even looks like that of an ant in some things [22]. Thus, we may suppose that system intelligence is the consequence of the system's purpose or predestination to use in practice. The approach does not assume environment modelling, unlike [17].

In this paper, first, we refine the results of [24], [25][†] and develop the approaches mentioned above. Our approach is based on the idea that one can imagine the process of studying the environment as the execution of parallel processes of detecting and recognizing various objects (i.e., processes to achieve them) in the environment. A tensor multiplication in a linear logic corresponds to these parallel processes. The logic is modelled in some game category [28] (first mention in [1], [2]). Thus, the process of revealing and achieving objects by the intelligent system in the environment can be described as a game. This technique allows us to give a natural description of concurrency in such an object recognition process and to gain the reward expressions from a construct associated with the logic. These expressions are consistent with our understanding of the meaning of the reward. Position rewards in the game are represented by some sets which define the information about these objects. The process of obtaining information is just the process of knowing the environment.

Thus, the main problems that we will explore in this paper are the recognition of interesting objects in the environment and the method for estimating information about them to obtain the game rewards. Usually, such a system has a pre-existing set of such items and a subsystem that recognizes them. However, what to do in an unknown environment? We suppose in this paper that the robot behaves like a baby: it moves toward objects which have attracted its attention, chooses the most attractive item, and looks at it from all sides. Thus, such an artificial intelligence system should be able to highlight unknown objects in the environment which were not specified in its pre-existing object set. To do this, we suggest using Biederman's theory of image recognition by humans [4]. According to the theory, people recognize object types by schemes that consist of geometrical primitives. Therefore, we can initially train the system's neural network to recognize only these primitives. Then, we can build their incidence matrixes for images of environmental objects[‡]. Thus, unknown objects (combinations of primitives in the environment) with the same set of primitives and the same incidence matrix would belong to the same object type. After that, these objects may be saved in a database.

This approach is based on the idea that the robotic system must go through the whole learning process as a real person, from childhood to adulthood to become an intellectual one. During the process, it must explore objects of the outside world, recognize and classify them, and evaluate their utility based on intent to use the system in practice. The first step is to teach the system to recognize objects' types and assign them names (classify) in its internal database.

When the robot moves, such an unknown object becomes more discernible, and the set of its primitives obtains new elements. Thus, we get an increment of information about the object. When this thing is studied from all sides, and we no longer have an increase in information, the process of the object cognition stops. Therefore, we may take these sets of geometrical primitives as the rewards in the game that describes the system movement in the environment. Similarly, while the robot is moving toward a possibly known object, the

[†]We give the improved notion of the payoff function in the game category used, adapt proofs of Propositions 2.1 and 2.2 to this case, and clarify the whole construction.

[‡]We experimentally demonstrate in the paper that such a trained network recognizes the primitives in real objects quite well. All experiments with incidence matrixes are left for the future.

uncertainty of its identification diminishes, and the set of possible identification variants may be taken as the game anti-reward (which decreases during the game in this case).

Our method of object recognition belongs to the class of local-based recognition methods [33]. However, these methods may consider any kind of local features that describe a local area of the object, instead of having appearance-based parts [34]. This technique requires a dataset composed of training images to select parts of the thing to be later recognized. Before training, the method itself does not allow you to distinguish between multiple classes of objects. A similar technique is used in [11] with other methods of part selection and another learning algorithm. A more complicated approach is described in [12], [13], [14], [29], [36] which uses the joint model of the parts to facilitate the detection of the features. Each part encodes local visual properties of the object, and the deformable configuration is characterized by a tree structure of connections between certain pairs of components. However, this method also requires a set of images to train part selection and to create an object class template.

On the contrary, we know features in advance in our method, and can discriminate between different object classes immediately, without any training. Also, in the first steps, we do not determine these classes precisely. The information obtained is increased from step to step up to the end. After that, we get an item in the database of such classes as a set which consist of all the object schemes at once. The item represents from all sides a comprehensive image of the corresponding object at the end of the process of learning it. Subsequently, a graphical skeleton of these geon schemes could be generated, which matches the item, and a neural network can be trained to store such a 3D type as a new object as in [8] (see Sec. 4).

The idea of the latter image recognition approach [8] is to place items of a typical shape in a different orientation in a virtual environment and to get their images. These objects are generated from some models, and the item in our database can serve as such a model. The position and the appearance of each object in each perspective are known immediately without additional calculations. In our article, we initially have such images of basic geometric primitives only. In the process of the environment cognition, we could add more intricate object schemes in different orientations, built from primitives, as a new item in the memory. Initially, this memory is the database; afterwards, these items can be stored in the network.

The second difference of our method from the others is that we store shape information in a static incidence matrix and use neither a probabilistic method for this purpose nor the procedure of minimizing the energy of the constellation of parts.

The paper is organized as follows. Section 2 presents the backgrounds necessary for the understanding of the results. We adapt the theory of [28] to the paper purposes in Subsection 2.2. Section 3 represents an updated variant of a formal description of intelligent system behaviour from [24], [25]. Section 4 represents the use of a neural network to obtain the information rewards in the game that describes the system behaviour. In Sec. 5 we discuss the reward valuations. In Sec.6, we conclude the paper.

2. BACKGROUNDS

2.1. Lattices

We suppose visible objects and, hence, the game rewards form different lattices. Thus, we start with them.

Definition 2.1:
*The **partially-ordered set** P is a set with such a binary relation $x \leqslant y$ on its elements, that for all $x, y, z \in P$ the following relationships hold:*

- *$x \leqslant x$ (reflexivity);*
- *if $x \leqslant y$ and $y \leqslant x$, then $x = y$ (anti-symmetry);*

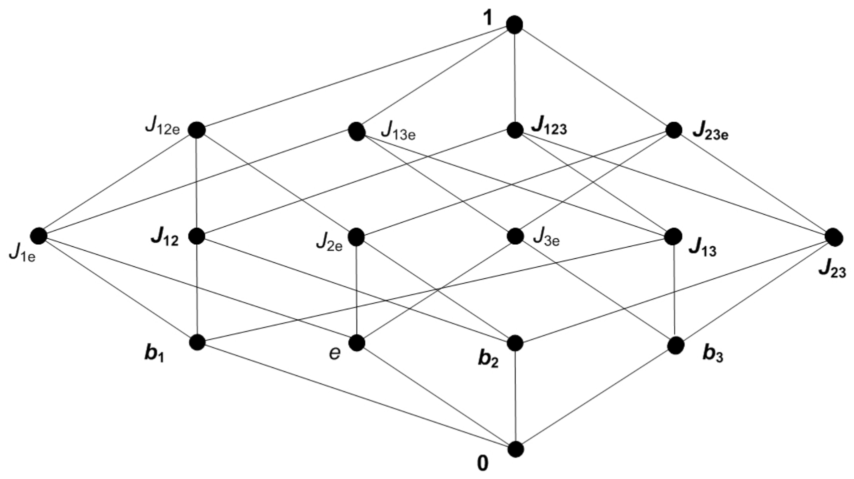

Fig. 2.1. A lattice example. An element J_{12} is the join of b_1 and b_2, J_{12e} is the join of b_1, b_2, and e, and etc. Elements $\{b_1, b_2, b_3, e\}$ are the set of generators.

- *if $x \leqslant y$ and $y \leqslant z$, then $x \leqslant z$ (transitivity).*

The definition means that in the partially-ordered set, not all elements are compared with each other. This property distinguishes these sets from linearly-ordered ones, i.e., from numeric sets which are ordered by a norm. Thus, the elements of the partially-ordered set are the objects of a more general nature than numbers. In the partially-ordered set diagram, the greater the element (i.e., vertex, node), the higher it lies, and the elements that are compared with each other lie in the same path from a bottom element to a top one. An example of a partially-ordered set diagram is represented in Fig. 2.1 which is also a lattice diagram.

Definition 2.2:
*The **upper bound** of a subset $X \subset P$ in a partially-ordered set P is the element $a \in P$ such that $x \leqslant a$ for all $x \in X$.*

The supremum or *join* is the smallest subset X upper bound. The infimum or *meet* defines dually as the greatest element $a \in P$ such that $a \leqslant x$ for all $x \in X$.

Definition 2.3:
*A **lattice** is a partially-ordered set, in which every two elements have their meet, denoted by $x \wedge y$, and join, denoted by $x \vee y$.*

In the lattice diagram, the elements join is the nearest upper element to both of them, and the meet is the nearest lower one to both.

Definition 2.4:
***Generators** are such elements that generate all other elements by joins and meets (They are b_1, e, b_2, b_3 in Fig. 1).*

Definition 2.5:
*The lattice is referred to as a **complete** lattice if its arbitrary subsets have the join and the meet.*

Thus, any complete lattice has the biggest element 1, and the smallest one 0 and every finite lattice is complete [5].

If we take such a lattice as a scale of truth values in multi-valued logic, then the largest element will correspond to complete truth (1), and the smallest to complete falsehood (0). Intermediate elements will correspond to partial truth in the same way as in fuzzy logic, partial truth is estimated by elements of the segment $[0, 1]$.

In logics with such a scale of truth values, the implication can be determined through the multiplication of lattice elements (residual logics), or internally, only from lattice operations.

Definition 2.6:
*A lattice that has internal implications is called a **Brower** lattice.*

In such a lattice, the implication $c = a \Rightarrow b$ is defined as the largest $c : a \wedge b = a \wedge c$.

Definition 2.7:
*The implication $\neg a = a \Rightarrow 0$ is called the **pseudo-complement**.*

Distribution laws for union and intersection are satisfied in the Brower lattices. The converse is true only for finite lattices.

2.2. Game Semantics

Definition 2.8:
*A **Conway game** is defined as a rooted graph with vertices* V *as the game positions and edges $E \subset V \times V$ as the game moves. Each edge has a **polarity** ± 1 which depends on whether it is the Proponent or the Opponent move.*

Definition 2.9:
*A trajectory or a **play** is some path from the graph root $*$. The path is **alternated** if the adjacent edges are of different polarities.*

Definition 2.10:
*A **strategy** σ of a Conway game is defined as a non-empty set of alternated plays (paths) of even length. They start from the Opponent move, closed up to the prefix of even length, i.e., for all plays* s *and all moves* m, n, *$s \cdot m \cdot n \in \sigma$ implies $s \in \sigma$, and determined. Determinism means that two different paths with a common prefix should coincide, i.e., for all plays* s, *and all moves* m, n, *and* n', *$s \cdot m \cdot n \in \sigma$, and $s \cdot m \cdot n' \in \sigma$ implies $n = n'$.*

Definition 2.11:
A dual play $X^{\perp}$ is obtained from the play X *by reversing the polarity of moves.*

Definition 2.12:
The tensor product $X \otimes Y$ of two Conway games X *and* Y *is the product of the two underlying graphs, i.e., its positions $x \otimes y$ are vertices $V_{X \otimes Y} = V_X \times V_Y$ with the game root $*_{X \otimes Y} = *_X \times *_Y$, its moves are $x \otimes y \to \begin{cases} z \otimes y; x \to z \text{ in } X \\ x \otimes z; y \to z \text{ in } Y \end{cases}$. The polarity of a move in $X \otimes Y$ is inherited from the polarity of the underlying move in* X *or* Y.

Generalized linear logic is modelled in the category **Conw** of such games [28]. The category objects are Conway games, and morphisms $X \to Y$ are strategies in $X^{\perp} \wp Y$. The definition of the categorical construction of the operation $\wp$, which is dual to tensor $\otimes$, is not discussed here for simplicity because this is not essential for our description. It is enough to mention that on game graphs, these two operations are the same, so in [28], they are not even distinguished. The morphism composition and identity morphism are apparent [28]. We do not need the construction of the linear logic here, except the notion of linear implication.

Definition 2.13:
[28], [20] The linear implications $X \multimap Y$ in the category are defined as
$$X \multimap Y = X^{\perp} \wp Y$$

since the category is symmetric monoidal closed (thus, we may define linear logic and the implication in the category). Thus, morphisms $X \to Y$ are strategies in the linear implication $X \multimap Y$.

A Conway game *X* with a payoff is the game with an additional weight $k_X = \{1, 1/2, 0\}$ in each vertex [28]. The weight depends on whether the position is winning or not. In the tensor production and implication, these weights obey the usual rules of conjunction and implication for such truth values scale [28]. Thus, the Conway's payoff game $X \otimes Y$ is defined as the underlying Conway game $X \otimes Y$, equipped with the payoff function $k_{X \otimes Y}(x \otimes y) = k_X(x) \wedge k_Y(y)$. The Conway's payoff game $X \multimap Y$ is defined as the underlying Conway game $X \multimap Y$, equipped with the payoff function $k_{X \multimap Y}(x \multimap y) = k_X(x) \Rightarrow k_Y(y)$. A strategy σ on a Conway's payoff game X is winning when every play $s : x \mapsto y$ in the strategy ends in a winning position y, i.e., in a position of payoff 1/2 or 1.

It is possible to prove that the categorical construction is conserved if the weights' numbers are replaced with some sets which form a Brouwer lattice. Also, the operations of [28] must be replaced with lattice operations. Thus, we may declare by definition

$$k_{X \otimes Y}(x \otimes y) = k_X(x) \wedge k_Y(y) \tag{2.1}$$

with lattice $\wedge$,

$$k_{X^{\perp} \wp Y}(x^{\perp} \wp y) = \neg k_X(x) \vee k_Y(y) \tag{2.2}$$

with lattice $\vee$, and

$$k_{X \multimap Y}(x \multimap y) = k_X(x) \Rightarrow k_Y(y) \tag{2.3}$$

where $\Rightarrow$ is now the lattice implication. Both of the last two definitions are suitable as the payoff function of the game $X \multimap Y$. The first definition is a particular case of the second one in Boolean lattices, and it may be used in practice as more convenient.

However, we must emphasize that these payoff definitions are purely voluntaristic. They do not follow from any mathematical construction. The unique reason to introduce them is interpreting the tensor production $\otimes$ as a multiplicative conjunction and the co-tensor production $\wp$ as multiplicative disjunction in linear logic. Hence, we may invert the payoff definitions 2.1 and 2.2, since operations $\otimes$ and $\wp$ are mutually dual as $\wedge$ and $\vee$. Thus, we introduce:

$$k_{X \otimes Y}(x \otimes y) = k_X(x) \vee k_Y(y) \tag{2.4}$$

$$k_{X^{\perp} \wp Y}(x^{\perp} \wp y) = \neg k_X(x) \wedge k_Y(y). \tag{2.5}$$

We will use these definitions in Sec. 3 since we interpret in the section the fulfilling of parallel processes in the tensor production as the processes' join in the corresponding lattice. We will prove that the categorical construction is also conserved for these definitions.

The larger the set is associated with a position, the more preferable it is. We suppose the existence of a universal set containing all the others. Thus, all such estimation sets form a complete lattice.

Definition 2.14:
We call a strategy σ on a Conway's payoff game X *with position estimations in a lattice as* winning *if every play $s : x \mapsto y$ in the strategy ends in a position y of payoff in the lattice which is different from 0.*

Let us note here, that linear implication $X \multimap Z$ means the process (game) X consuming and the process (game) Z obtaining [16]. Therefore, the payoff $k_Z(z)$ may not be 0 in the end position of the winning strategy σ of the game $X \multimap Z$, because we must not obtain a process (game) Z ended in 0 in the winning case. However, this restriction is unnecessary in the case of definitions (2.4) and (2.5). We show now that such defined winning strategies do compose as in [28].

Proposition 2.1:
The strategy $\rho \circ \sigma : X \multimap Z$ is winning when the two strategies $\sigma : X \multimap Y$ and $\rho : Y \multimap Z$ are winning.

Proof
Let us consider first the proof for (2.3). It is known that strategies do compose [28]. Thus, it

is sufficient to check the winning condition. We should observe that the composition of two winning positions is winning:

$$k_X(x) \Rightarrow k_Y(y) > 0,\ and\ k_Y(y) > 0,\ i.e.,\ x \multimap y\ is\ winning;$$
$$k_Y(y) \Rightarrow k_Z(z) > 0,\ and\ k_Z(z) > 0,\ i.e.,\ y \multimap z\ is\ winning;$$
$$implies\ k_X(x) \Rightarrow k_Z(z) > 0,\ i.e.,\ x \multimap z\ is\ winning.$$

However, it is evident, since $k_X(x) \Rightarrow k_Z(z) \geqslant k_Z(z) > 0$.

The proof for (2.2): it is evident that in the case of $k_{X\multimap Y}(x \multimap y) \equiv k_{X^\perp \wp Y}(x^\perp \wp y) = \neg k_X(x) \vee k_Y(y)$ and $k_Y(y) > 0$ winning strategies do compose.

The proof for (2.5): it is evident that in the case of the winning strategy with $k_{X\multimap Y}(x \multimap y) \equiv k_{X^\perp \wp Y}(x^\perp \wp y) = \neg k_X(x) \wedge k_Y(y)$, $\neg k_X(x) > 0$ and $k_Y(y) > 0$, thus winning strategies do compose. □

Definition 2.15:
*Let us, **SetPayoff** is a category whose objects are Conway's payoff games, in which position weights take values in a lattice, and morphisms $X \to Y$ are winning strategies in $X \multimap Y$.*

Proposition 2.2:
*The category **SetPayoff** is symmetric monoidal closed.*

Proof
The proof for (2.1) and (2.3): the category of Conway games is symmetric monoidal close [28]. Therefore, it is sufficient to check if $(k_X(x) \wedge k_Y(y)) \Rightarrow k_Z(z) = k_X(x) \Rightarrow (k_Y(y) \Rightarrow k_Z(z))$ for all positions. But this formula is valid in Heyting algebras (i.e., in Brouwer lattices).

The proof for (2.1) and (2.2): $\neg(k_X(x) \wedge k_Y(y)) \vee k_Z(z) = \neg k_X(x) \vee (\neg(k_Y(y) \vee k_Z(z))$ in Brouwer lattices.

The proof for (2.4) and (2.5): $\neg(k_X(x) \vee k_Y(y)) \wedge k_Z(z) = \neg k_X(x) \wedge (\neg(k_Y(y) \wedge k_Z(z))$ in Brouwer lattices. □

Thus, the symmetric monoidal closed categorical construction for Conway's payoff games from [28] is conserved for lattice payoffs, and morphisms $X \to Y$ are strategies (in this case winning) in the linear implication $X \multimap Y$ as in Conway's games'category.

2.3. Biederman Geons

Irving Biederman has proposed in [4] a theory of image recognition by humans, in which an image is segmented into a set of geometric primitives, such as blocks, cylinders, wedges, and cones. The collection of the components, called geons, is rather limited ($N \leqslant 36$) and can be derived from easily detectable properties of edges: curvature, collinearity, parallelism, and convergence. The detection of these properties is invariant over viewing position and image quality, and thus allows objects to be perceived in different positions and in the case of noised image. The experiments of [4] showed low errors in object naming of such geon schemes of different objects. Some of such simple geon schemes are represented in Fig. 2.2. The approach is similar to speech perception: in both cases, we can code tens of thousands of objects by mapping the input onto a limited number of primitives and then using a representative system to build free combinations of these primitives. However, this method does not allow us to distinguish images inside one object type. To do this, we should use other methods.

3. SYSTEM BEHAVIOUR DESCRIPTION

We consider the cognition process as a game in which a system (a robot) investigates an environment, i.e., the system obtains information about the environment objects in the form of geon sets.

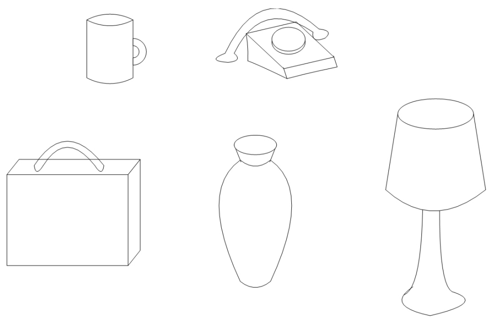

Fig. 2.2. Examples of geon schemes of simple objects.

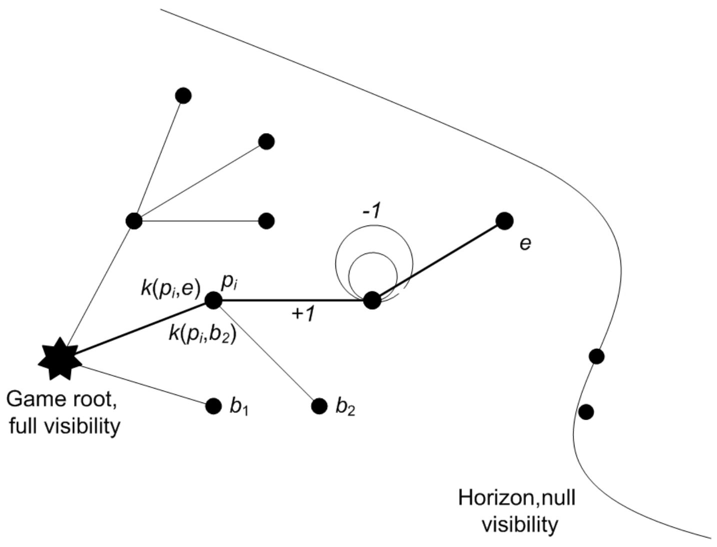

Fig. 3.3. An example of a winning strategy of a game and the visibility horizon. The bold line shows the resulting play (one of the winning strategy) with the rewards $k(p_i, b_j)$ in the positions p_i of the process of reaching objects b_j and with polarities ± 1. e and b_j are objects of interest $i, j = 1, 2, ...$. The environment moves — obtaining information by the robot — are depicted by circles, since they are not real moves.

Supposition 3.1:
It is supposed that the robot investigates the environment visible up to some horizon in each direction (Fig. 3.3) and builds images of the observed objects.

Supposition 3.2:
It is supposed that objects' degrees of attractiveness guide the robot to behave in a certain way: it investigates things in the environment which have attracted its attention, and builds their images.

The theory of attention is discussed, e.g., in [15]. "Attention is defined as a concentrated mental activity. In general, we can think of attention as a form of mental activity or energy that is distributed among alternative information sources" [15]. Both general classes of theories that attempt to explain attention — bottleneck theories and capacity theories — explain *how* attention selects information sources. Still, they tell us nothing about *why* exactly it makes the selection. We suppose the system has some pre-existing preferences to choose the sources. In the simplest cases, the choice can be based on the detection of areas of difference in hue, contrast, degree of distance from the observation point, etc. [7] In more complicated cases, the system can have some frames or gestalt images. Thus, we suppose that the system mission or purpose for practical use determines the attention preferences.

Then, the system should distribute the objects according to the degree of attraction during the investigation. Thus, some objects are more attractive, some of them less attractive, and some cannot be compared in terms of attractiveness. Therefore, we get a partial order at the object set. It is supposed that the set has a bottom element, i.e., the element of zero interest and the top element, i.e., object of greatest interest, which is the most attractive element. The latter is the lattice join of all its elements (see Supposition 3.3). Other elements may be represented as some joins (combinations) or meets in the case when an attractive object is some part of another one. Therefore, the system builds a complete lattice of the environment objects' images, their combinations, and allotments (e.g., Fig. 2.1 for some objects from Fig. 3.3). Such a lattice may already exist; thus, the attractiveness of new objects may be combined with the attractiveness of pre-existing ones. Many simple biological systems, e.g., ants colonies, possibly have such a pre-built lattice of objects with which they can deal. Perhaps the partial order in the lattice can be dynamically changed for more complicated systems.

In any case, we assume that the lattice of the environment object images can dynamically change due to the fact that the system does not see everything around at every moment of time. The system looks into a sector and builds images of objects that lie inside it.

Supposition 3.3:
It is supposed that the preferable behaviour of the system is to achieve all its goals — i.e., to investigate and recognize all interesting objects in the environment at a given time, i.e., to get maximum information.

We identify the system goals (where "goals" mean interesting objects in the environment) and the processes of achieving these goals. Thus, the most preferable behaviour variant corresponds to the top lattice element 1. And the bottom element 0 (zero) corresponds to complete inactivity and to the least significant behaviour variant. All the estimations may be considered as partially true truth values. Thus, we can say that the more essential the system behaviour is, the truer it is. When it is not possible to determine the most crucial estimation for a behaviour due to the order being only partial, some additional methods may be used to select the optimal variant [24].

The process of moving a robot in the space of the environment, i.e., the cognitive process of studying the environment by a robotic system, can be represented as a game. During the game, the environment informs the system about (partially) visible objects at each step, and the system estimates the information and formes the position reward. In the next paragraph, we will see that the rewards are not numbers, but sets of geometric primitives which make up the objects. Certainly, the environment has no agency of its own, and it is passive as a player, thus, this is all very reminiscent of a reinforcement learning approach [32]. However, using the construction of the category of Conway games allows a natural description of the process of parallel exploration and achievement of various objects before choosing one of them. Moreover, we are interested not so much in the recognition of objects as in their cognition. The main thing is to get as much information as possible about an individual object, and not teach the robot any specific behavior. Also, this information as the rewards is represented by sets, not by numbers. Therefore, usual methods of reinforcement learning do not fit.

We regard the environment as the Opponent and the system as the Proponent to use the categorical construction of Sec. 2.3. The Proponent moves from one position in the environment to another, using the information to achieve his goals, i.e., environment objects. The more fortunate the position, i.e., the larger the reward, the more precise information the environment provides about the item in the position. Thus, the completely winning position is the last game step of the Proponent, in which it can get no more information. The robot is initially placed in the configuration space (environment) at the root * of the system game A with the system goal/object lattice M_s partially ordered according to robot attention preferences.

Game *A* represents the ***possible*** robot's moves in the environment. But the ***real*** trajectory or play is chosen from the requirement of the maximum total reward for a position along the projected path ending in a winning position (in this position the reward cannot be improved). The system movement in the environment is estimated according to this criterion with the reward $k(p_i, b_j)$ in the position p_i of the process of reaching the object b_j.

It may happen that the system initially does not intend to achieve any goal and moves according to the criterion of optimal movement in the absence of goals. Thus, the system has a goal (task) a of free movement in the environment, which should be included in the lattice M_s. Hence, such free movement must have its own value. We do not discuss here the optimality criterion, since the robot or agent systems' designers may suggest such ones according to their needs in a specific environment. However, we assume that the rewards for free movement in the corresponding game are also estimated by some sets. These sets are of the same nature as the sets which correspond to the rewards in the game of achieving goals. Therefore, the rewards of the free movement are some sets of geometric primitives that are placed in the environment but are not treated as objects of interest.

Thus, we suppose the system sees different combinations of geometric primitives in the environment. The primitives (geons) which are located in one place make up an object scheme. These schemes may have joins and meets combining other such schemes (which may no longer be in one place). Hence, such primitives' sets form a lattice. The lattice is complete since it contains any joins and meets. This lattice is different from the lattice M_s of the system goals/objects. The lattice M_s contains only a part of all objects, and it is partially ordered according to attention preferences. The primitives' lattice contains all visible things and their combinations, and it is ordered according to the relation of the set inclusion. We demand the geons' lattice is Brouwer (in particular, it must be distributive); thus, it includes all that is necessary for these joins and meets even when they are not evident.

Let us *n* objects $b_1 ... b_n$ are discovered in the environment by the system with information about them $k(p_i, b_j)$ in positions p_i of the game *A* (Fig. 3.3). The rewards $k(p_i, b_j)$ take values in the geons' lattice. Only k goals from these n's ones may be chosen due to limited system opportunities. Game *A* corresponds to the process of achieving the free movement goal *a*. Then, a winning strategy of the game $A \multimap B_1 \otimes ... \otimes B_k$ (i.e., a set of winning plays) defines a transition (morphism) from the game *A* to this new game $A' = B_1 \otimes ... \otimes B_k$ of moving and achieving goals $b_1 ... b_k$ in the games $B_1 ... B_k$. It is supposed that the system can achieve several objects in parallel consistently reducing their number up to the moment when only one object rests to be chosen. Thus, the game A' corresponds to parallel processes of reaching those *k* goals from discovered *n*'s ones, which may be better achieved in the next sense.

It is reasonable to choose the trajectory (one play from the winning strategy) from the requirement to maximize the reward along the path within the visibility horizon, in the following way (using (2.4), (2.5) in contrast to [24], [26])[§]:

$$k_{play}^{A \multimap B_1 \otimes ... \otimes B_k}(a \multimap b_1 \otimes ... \otimes b_k) \equiv k_{play}^{A^{\perp} \wp B_1 \otimes ... \otimes B_k} =$$
$$= \max_{plays}[\bigcup_{play} \neg k^A(a) \wedge (k^{B_1}(b_1) \vee ... \vee k^{B_k}(b_k))] \quad (3.6)$$

Here, the reward $k_{play}^{A \multimap B_1 \otimes ... \otimes B_k} = k_{play}^{A^{\perp} \wp B_1 \otimes ... \otimes B_k}$ corresponds to that process of achieving *k* goals that has the greatest priority (i.e., the highest truth value) in the system goal/object

[§]We may also use formulas (2.3) and (2.1) for rewards:

$$k_{play}^{A \multimap B_1 \otimes ... \otimes B_k} = \max_{plays} \bigcup_{play} [k^A \Rightarrow (k^{B_1} \wedge ... \wedge k^{B_k})],$$

i.e., with meets in the rewards for games $B_i, i = 1...k$. But, this form contradicts our reward sense (Sec. 5). However, the implication calculation is no more difficult [27], hence, the form could also be used with some another reward definition.

lattice (at the current time). Thus, these *k* processes are the most important parallel ones from the viewpoint of the system goal lattice. The priority is maximal among all possible parallel processes of reaching *n* discovered objects[¶].

The maximum in (3.6) is taken among all possible plays, and it joins the rewards along these plays (i.e., trajectories) in games $A^{\perp}$, B_1, ... and B_k. Thus, information about all objects b_i and their system images is demanded to maximize along the resultant path up to the moment when all the images together may not be able to improve. Then, the total number of chosen objects should be decreased by the same method: the system should pick those l objects with $l < k$, the processes of parallel achievement of which have the highest estimation in the system goal lattice. And so on, up to the moment, when perhaps only one goal remains. However, some external method of the path calculation may suggest an objects' traversal path to obtain the maximum cumulative reward from as many angles as possible (see discussion in Sec. 5).

The meaning of formula (3.6) is that, following the semantics of linear implication, the system moves from the execution of the process of free movement A to the processes of achieving goals B_j. The gain is calculated within the strategy (i.e., a set of pathes) of game $A^{\perp}\wp B$ (with simplified notation). At the same time, information about targets $k^{B_j}(b_j)$ is maximized, and information about the free movement of $k^A(a)$ is minimized (since pseudo-complements $\neg k^A(a)$ are maximized). Such k and $\neg k$ can be considered as arguments and counterarguments for the corresponding movement following the ideas of the JSM method of plausible inference [3]: the more information we have about the object, the stronger the arguments for the transition to achieving it. Also, the stronger the arguments against free movement (i.e., the less information we have about the possibility of such a movement), the stronger the arguments for moving from free movement to achieving a goal.

4. OBJECT RECOGNITION

Thus, we come to the main problem: how to select (recognize) objects in the environment and how to evaluate the amount of information about them, i.e., the rewards? We offer a solution to the latter problem in Sec. 5 "Discussion of System Movement and Reward Valuations", and we discuss below the former one.

We suggest to decompose an environment image to Biederman's geons (paragraph "Biederman Geons" 2.3 in Sec. 2). Then, during the cognition process, the system obtains such images from different camera angles, and gets a tuple of sets of geons localized in one place. These sets with their geon incidence matrixes constitute one type of environment objects. The incidence matrix for each geon pair indicates their facets which intersect (see paragraph "Geon Recognition" 4.4 below). If the system recognizes such a set in future, it will know the possible object type.

To perform the programme, we need to automatically train the neural network that will be used for recognition, first only these geons (this is done in the paper), and then such sets of geons (which we suppose to perform later). Thus, we consider first an automatic generation of training samples in a virtual environment [8], [9].

4.1. Generation of training samples

The presence of a high-quality training set largely determines the efficiency of machine learning algorithms. It is worth mentioning that when preparing training samples, you should pay attention not only to the volume of data, but also to such things as the balance of classes

[¶]We consider here the join of several goals in the lattice M_s as the process of their possibly parallel achieving (like in [22]), though, in [24] a linear logic structure on the goal lattice was used in this case. We did so because it is hard making sense of lattice elements multiplication in linear logic.

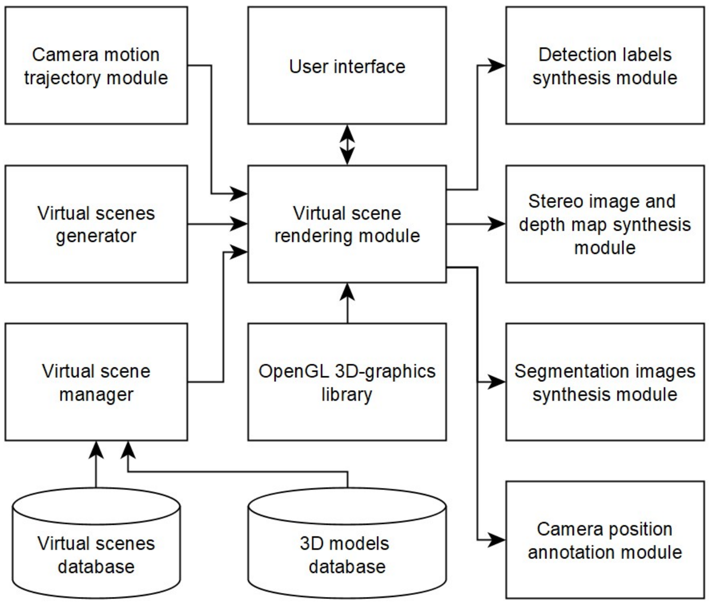
Camera motion trajectory module
User interface
Detection labels synthesis module
Virtual scenes generator
Virtual scene rendering module
Stereo image and depth map synthesis module
Virtual scene manager
OpenGL 3D-graphics library
Segmentation images synthesis module
Virtual scenes database
3D models database
Camera position annotation module

Fig. 4.4. The structure of the dataset synthesis software (DSS).

and the order of their sequence. The data must contain a comparable amount of instances for each class, and must be mixed. It is advisable to include such data in the training set that is as close as possible to the conditions of further use of the neural network.

In this study, we propose a technology of synthesis of training sets based on the use of three-dimensional graphics (OpenGL library [19]) within the developed software-algorithmic complex (Fig. 4.4).

The dataset synthesis software (DSS) [9] is based on the module for rendering images of virtual scenes. A virtual scene is a collection of three-dimensional objects of different categories, equipped with a description of their position, orientation and colour characteristics.

The input of the scene rendering module is the information about the geometric and textural features of the scene, received from the virtual scene manager according to the data stored in the file system. Alternatively, an instantly generated scene created with a virtual scene generator is used. Unlike the loaded scene, the positions and number of objects in the generated scene are usually randomly selected, which creates the required variety for generating training examples for a neural network. The coordinates of the camera are also transmitted to the input of the virtual scene rendering module during its movement along a given trajectory. This feature allows simulating the video image from the onboard camera to model navigation tasks of mobile robots. The output image of the scene rendering module is transmitted to the modules for generating training examples. Thus, e.g., the module for generating detection examples generates text descriptions containing the coordinates of target objects locations on the video frame. In turn, the module for analyzing the stereo image synthesizes a depth map corresponding to the observed visual scene. The segmentation module generates maps of pixel belonging to certain objects in the field of view of the camera. And finally, the camera position annotation module saves the camera transformation matrix at the current time in the training example.

4.2. *Training sets formation in problems of visual classification*

In accordance with tasks of visual image analysis, DSS allows you to generate training samples for solving the problems of visual classification, localization, segmentation and image depth evaluation. In addition, the virtual environment provides access to the exact position of the camera at successive times, which makes it possible to synthesize training samples to solve the problem of visual odometry.

There are two ways to create virtual scenes in the pre-rendering phase.

We use the following approach for the tasks of coarse tuning of neural network classifiers, when the mutual position of different objects is not important and, on the contrary, requires the greatest possible variety of objects moving around the scene.

In the approach, the number of N objects simultaneously observed in the scene is manually set or randomly selected. A list of random positions $p_i, i = 1...N$ for these objects is formed:

$$P = \{p_1, ..., p_N\}. \tag{4.7}$$

The situations of mutual penetration of objects are eliminated on the basis of the method of potential fields:

$$p_i' = p_i + min(d_{max}, \sum_{j=1, j \neq i}^{N} \eta_{ij} / \parallel p_j - p_i \parallel^2), \tag{4.8}$$

where $p_i' = \{x', y', z'\}$ is an updated object position; d_{max} is a maximum shift; η is the repulsion coefficient for the pair of objects i and j, $\parallel \bullet \parallel$ is the Euclidean norm.

Coefficient η is chosen for reasons of repulsion normalization. Let R_1 be the safety radius of the first object, R_2 the safety radius of the second object. Then the target distance between objects is $D = R_1 + R_2$. The condition for the empirical calculation η can be chosen as follows:

$$\eta_{ij} / \parallel (p_j - p_i) \parallel = 0.5 D_{ij}. \tag{4.9}$$

For the tasks of configuring neural network classifiers to solve specific application problems, an approach based on loading pre-prepared virtual scenes is used. In such a case, the variety of training examples is achieved not by changing the position of objects in the scene, but by changing the angle of observation when the camera moves along the specified trajectory. At the first stage of solving a specific applied problem of setting up a visual classifier, software is used that allows to form descriptions of virtual scenes in the following form:

$$W = \{o_1, ..., o_N\}; \tag{4.10}$$

where o_i is a programme structure that encapsulates the object's position, orientation, type and specifics of its visual appearance. Compatibility of virtual scenes' storage format is ensured to provide their correct loading in the virtual environment.

In the second stage, automatically generated scenes are loaded into the DSS: the text descriptions of the scenes are interpreted and the corresponding software representations are formed for the objects listed in the scene file. Next, reliable results of visual analysis are determined on the basis of a direct access to the properties of the loaded programme structures. Annotations containing the desired results of the analysis of the type and position of objects are saved together with the images in a directory on the disk for further tuning neural networks. Let us note that the first stage can be performed either using ready-made three-dimensional models as the elements of the set, or using procedurally synthesized models. The latter approach is preferable due to the unlimited possibilities for altering the parameters and, as a result, achieving variability in the appearance of these objects.

4.3. Procedural model generation

Procedurally generated objects are specified as a set of parameters, which, on the one hand, is more compact compared to the explicit enumeration of constituting geometric primitives in the three-dimensional model file. On the other hand, the set allows you to diversify the appearance of the object to form high quality training sets. In general, the generated object is

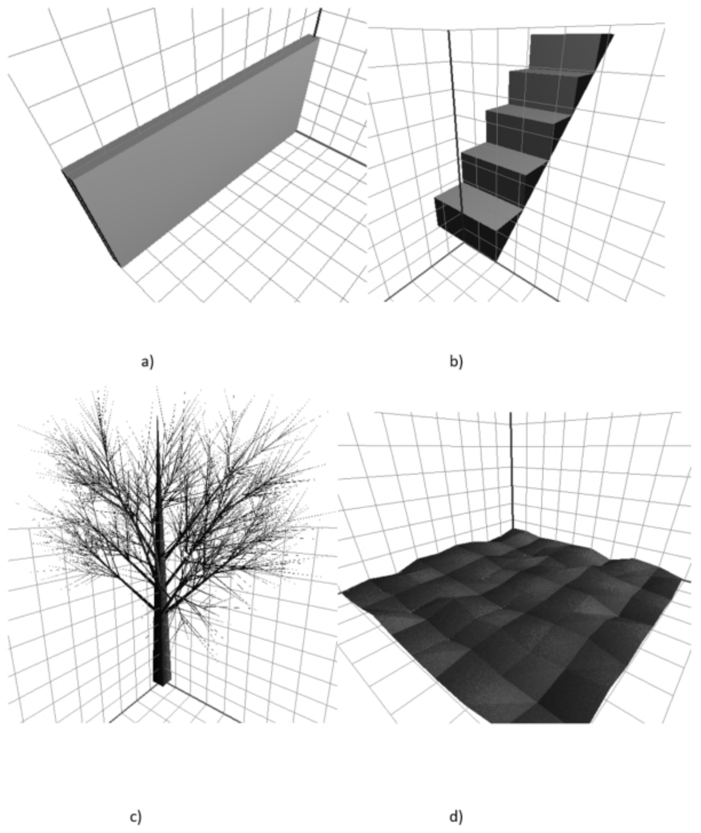

Fig. 4.5. Examples of automatically generated objects: a) wall; b) stairs; c) tree; d) a fragment of the landscape.

specified by the formula

$$o = \{l, p_1 ..., p_K\}, \tag{4.11}$$

where l is object class, $p_i,\ (i = 1...K)$ are parameters determined by an expert, which affect the geometric shape of the object through the relations embedded in the corresponding analytical model. Examples of procedurally synthesized objects used in modelling both indoor and outdoor areas are presented in Fig. 4.5. For example, to generate an object of type *"Tree"* (Fig. 4.5,c), the set (4.11) takes the form:

$$o = \{\text{“\textit{Tree}”}, h, k, n_b, n_l\}, \tag{4.12}$$

where h is the tree trunk height, k is a thickness coefficient, n_b is a number of branches at every level of the tree, n_l is a number of levels of recursive branching.

Each branch is formed as a typical tetrahedral pyramid, the displacement and rotation of which are set randomly within the limits allowed by the parent object (trunk or branch). Its length is selected in proportion to the distance from the positioning point of the branch to the final vertex of the parent object. This approach allows you to create a wide class of objects that look like real trees of different types. The same can be said about the other objects presented in Fig. 4.5.

The described possibility of procedural generation of objects can be used not only for the formation of training sets in the classification of visual images, but also when setting up vision systems that solve the problem of avoiding obstacles. The presence of comprehensive information about the geometry of the synthesized object enables optimal trajectory planning in the nearby space. Comparison of the obtained plan of movement with partial visual information available for the moving camera makes it possible to form a dataset with samples that capture implicit relationships when choosing an adequate movement of the robot for various observed situations in the external environment.

An additional increase in the number of training examples can be obtained using well-known methods of augmentation: shifting, rotating, reflecting, scaling, noising, blurring images, as well as a relatively new method of neural network data augmentation [37].

4.4. Geon Recognition

Now, we can flesh out the formula (4.11) for geons:

$$geon_i = \{id_i, \{(facet_j, line_j, deformed, curved)\}\}, \tag{4.13}$$

where “line” is the linear parameter of the j’s facet, which takes values $\{long, short\}$, “curved” is its curvature parameter, which takes values $\{yes, no\}$, and the parameter “deformed” takes values $\{no, bloat, depressed\}$. The facet set is: $\{top, bottom, front, back, left, right\}$ with the corresponding geometrical form for each value. All geons are modifications of a cylinder and a brick. Accordingly, each brick is a combination of rectangles (and trapeziums up to triangles) that have a long and a short size. The cylinder also has such an aspect ratio. The “line” parameter indicates this characteristic. Additionally, geons may be parameterized with scaling factors, e.g., $s_x = [0.5...2], s_y = [0.5...2], s_z = [0.5...2]$, and surface noise $F = [0...1]$, which allow us to diversify their visual appearance during the generation of the training images. We have used such a diversification in this paper.

Then, we get the following values for the incidence matrix of a geon pair:

$$M_{i,j} = \begin{cases} \{zone_j\}, \; geon_i \bigcap geon_j \neq \emptyset; \\ 0, \; geon_i \bigcap geon_j = \emptyset, \end{cases} \tag{4.14}$$

where the variable “$zone_j$” refers to j’s geon and takes the next values:

$$zone_j \in \{top, bottom, left, right, front, back\}_j. \tag{4.15}$$

The set of zones denotes those frame sides of j’s geon which intersect with the frame of i’s geon. We get corresponding frame sides of i’s geon in the $M_{j,i}$ element. Thus, we can reconstruct the whole geon scheme of the object in a certain foreshortening by its incidence matrix.

The incidence matrix and zone values can be obtained with fuzzy operations (especially, when the objects intersect by two or more zones) from the objects’ bounding box coordinates given by the used YOLO (“You Only Look Once”) neural network. Naturally, this is rather a rough object classification, and we offer to use it as the first simple variant.

At the first stage, these matrices can be storage in the system’s (robot’s) memory in order to compare them with the matrices of observed objects. Subsequently, we would like to build a 3D object by a corresponding incident matrix set and to train a network for such objects as we do in this paper for geons. Such a 3D object could be supplemented by a graph of geon connections to each other.

In this paper, as a first step, we have used the dataset synthesis software (DSS) [9] (Subsec. 4.1) to get a geon data set[‖], and trained a YOLO neural network on them with the resources [30], [31] with predefined training parameters there. At the network input, we have a raster RGB image, presented as an array of pixel brightness with a resolution of 416 * 416. At the output of the network, we obtain 169 cells, each of which contains 63 values: (4 coordinates of the frame $+1$ estimate of the presence of an object $+16$ geon classes) * 3 proportions of the frames = 63 values. Total: $169 * 63 = 10647$ values.

Fig. 4.6 — Fig. 4.11 show examples of decomposing real images on geons after the network training.

[‖]We used a set of only 16 main geons for our limited calculation capabilities.

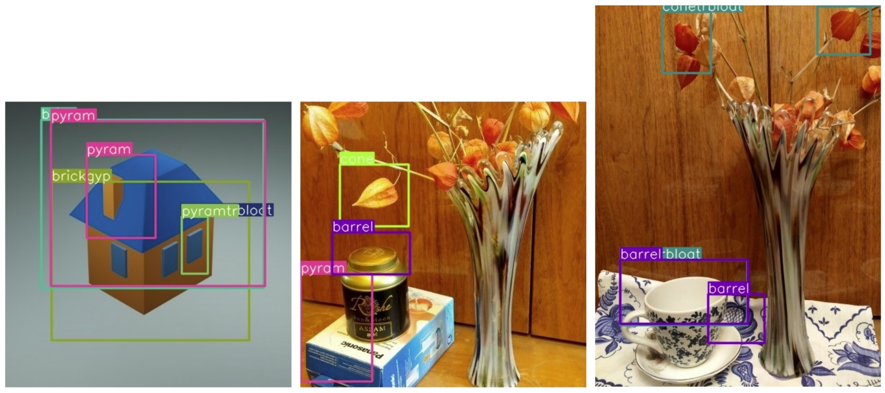

Fig. 4.6. An example of a simple image decomposition
in the case of automatically generated images in the training set.

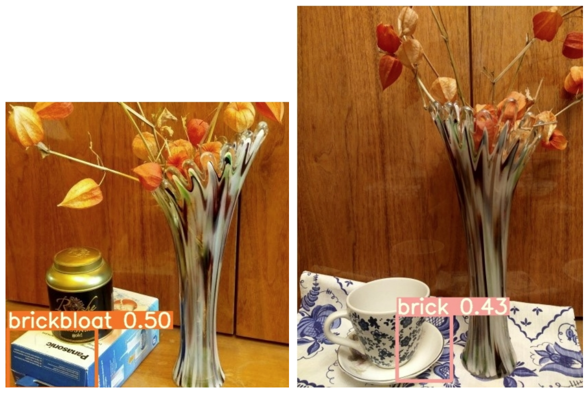

Fig. 4.7. An example of a simple image decomposition
in the case of manually marked up images in the training set.

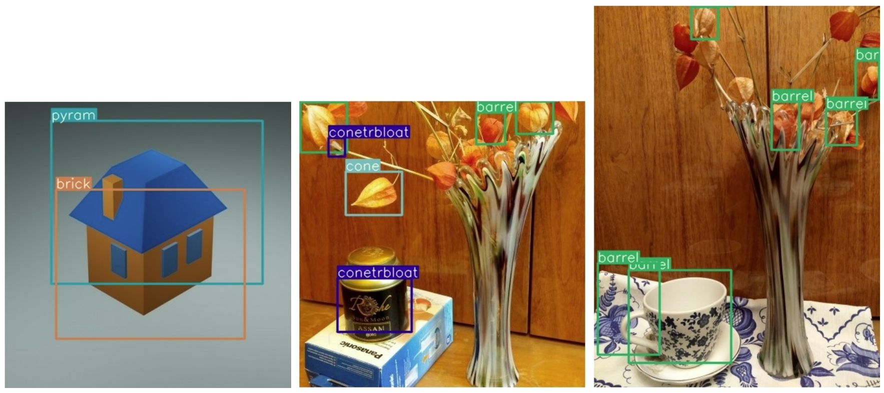

Fig. 4.8. An example of a simple image decomposition
in the case of automatically generated and manually marked up images
in the training set.

Here, you can see that only an automatically generated image set is not enough for good image recognition under limited training (Fig. 4.6, 4.9): we have used together three series of 150 images with 10, 20 and 30 geons respectively, with 10x augmentation (resource [35]

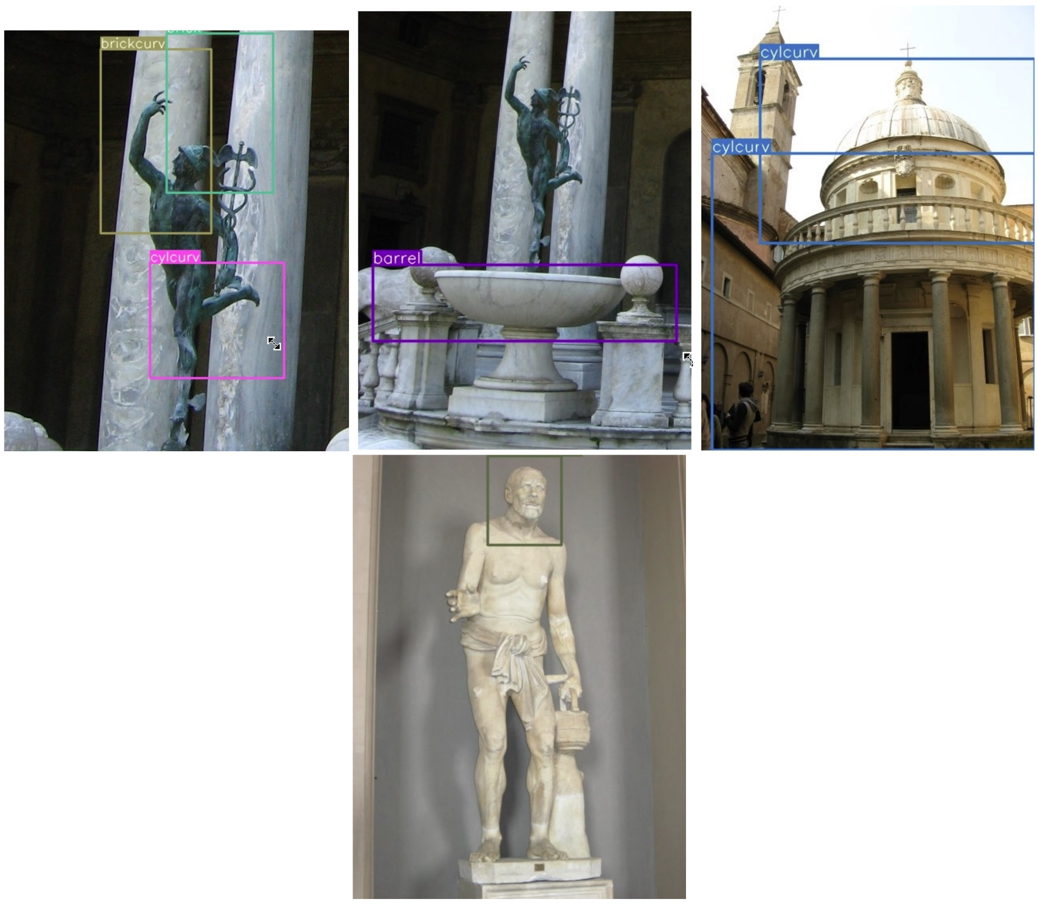

Fig. 4.9. An example of a complicated image decomposition
in the case of automatically generated images in the training set.

has been used). The manually marked up image set is also not good (Fig. 4.7, 4.10)[**], and only the combination of automatically generated images with a small part of the hand-marked ones gives an acceptable result (Fig. 4.11)[††] — geon recognition becomes much more exact and complete. We use here about 150 hand-marked images with the same augmentation.

5. DISCUSSION OF SYSTEM MOVEMENT AND REWARD VALUATIONS

Formula 3.6, which maximizes reward along the path, provides a mathematical justification for generally obvious behaviour. However, we can now clarify some points to use the formula in practice. Obviously, the system should move in such a manner as to increase all the rewards $\neg k^A(a)$ and $k^{B_i}(b_j)$ at some discrete time moments for the maximum possible number of goals. Thus, what amounts should we take as the objects' rewards? Clearly, the number of visible objects' geons may not increase as the system gets closer to the object. This number may even decrease with the foreshortening change. However, the information amount increases since we recognize the object more confidently and from different sides.

[**]An image is not presented in these image sets if it was not recognized in the corresponding case

[††]Authors are grateful to K. Rusakov for this advice

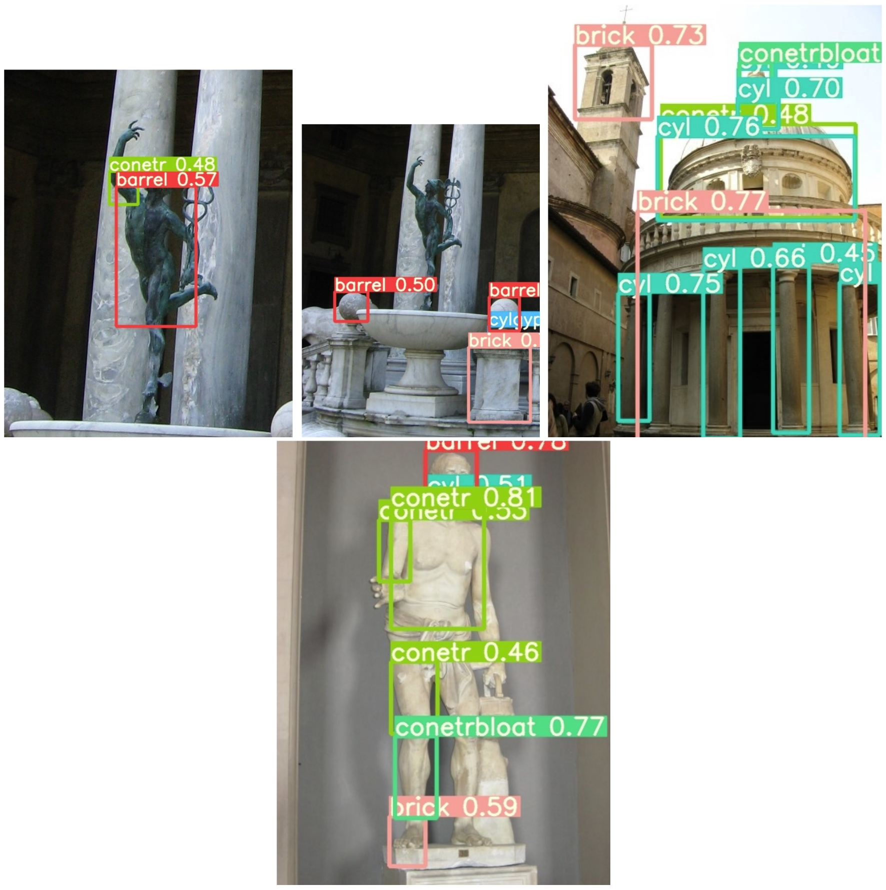

Fig. 4.10. An example of a complicated image decomposition
in the case of manually marked up images in the training set.

Hence, we use the set $\cup_j \{g_l\}_j$ of geons in geon schemes of goals b_j to be achieved as the anti-reward $\neg k^A(a)$ for the free movement. The larger the set, the higher the reward of the non-free movement and the lower the reward of the free movement. For $k^{B_i}(b_j)$ in the process of achieving goals, we use (possibly successively) two reward types :

— In the first type, the system approaches an object and sees it better and better. It tries to identify the object with some ones in the system database, and the uncertainty of its identification diminishes. At this stage, we use the anti-reward $\neg k^{B_i}(b_j)$ as if it is a possibly known object: the anti-reward is the set of possible recognition variants. When the system sees all visible geons $\{g_l\}_j$ of the thing b_j precisely, without any hesitation, then the anti-reward is minimal, and the reward $k^{B_i}(p_i, b_j)$ in position p_i is maximal (the object has been identified). Thus, the system should move in such a manner as to decrease the anti-reward set;

— In the second one, the system has no association in its database with the object. In this case, we take the union $\bigcup_{\substack{p_k \\ k\leqslant i}} \{g_l\}_j$ of geons' sets (remembered up to the position p_i) of the object b_j visible from different camera angles, as the reward $k^{B_i}(p_i, b_j)$. The geon sets are chosen to be localized in one place. This reward is maximal when the system has investigated the object from all sides and cannot get more information (i.e., cannot add new geons to the set corresponding to the item). Such a reward definition may demand external methods to decide that the object is investigated from all sides.

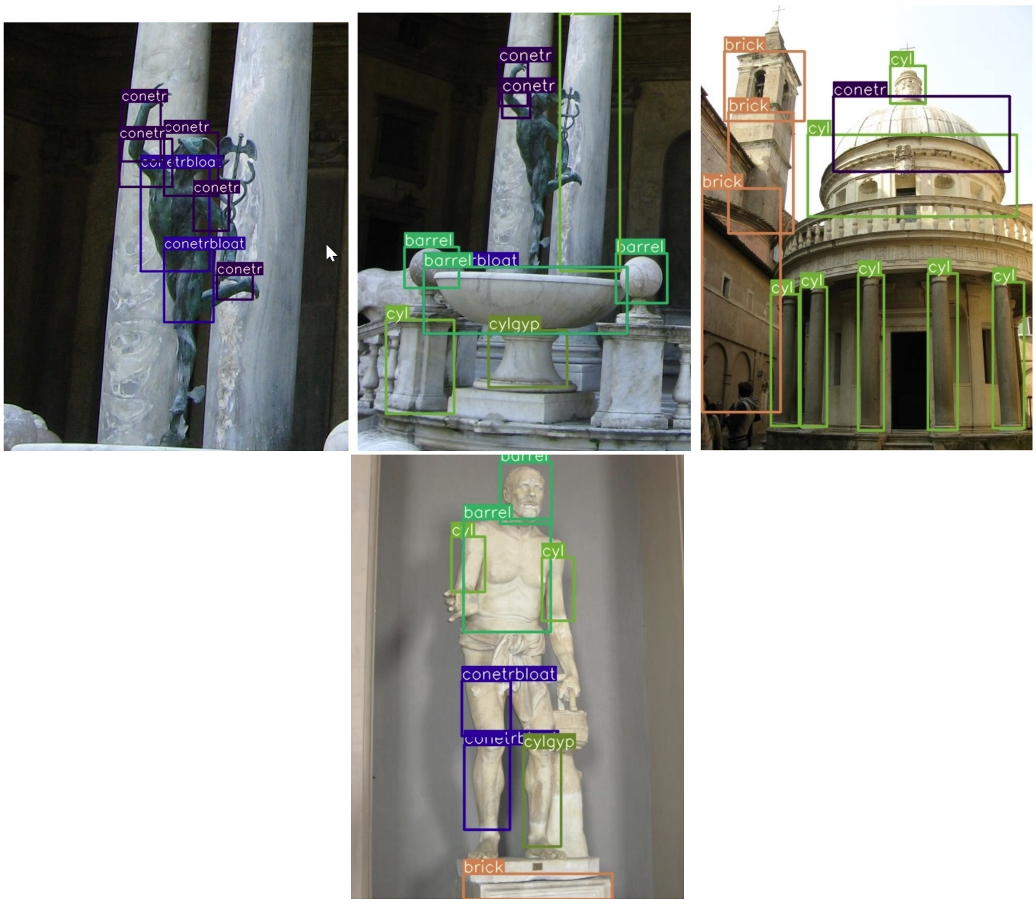

Fig. 4.11. An example of a complicated image decomposition
in the case of automatically generated and manually marked up images
in the training set.

Suppose, there are left only two objects of interest (goals) b_1 and b_2, and the reward of these two games B_1 and B_2 cannot be enlarged in parallel:

$$\bigcup_{play} \neg k^A(a) \wedge (k^{B_1}(b_1) \vee k^{B_2}(b_2)) = max. \tag{5.16}$$

The formula means that $k^A(a)$ cannot be diminished, as well as $k^{B_1}(b_1)$ and $k^{B_2}(b_2)$ cannot be increased, i.e., the set of known geons in b_1 and b_2 cannot be increased in parallel. Hence, we should choose the game and the play with only one object achieving. We do this according to our supposition to pick those objects which have greater assessments in the goals/objects lattice M_s (Sec. 3). This lattice is partially ordered by attention preferences. Thus, we reduce the number of goals that need to be achieved not from comparing their geon sets, but from comparing the degree of their attractiveness (attention preferences) for the robot.

Thus, formula 3.6 mathematically justifies the apparent general behaviour of the system, but not a specific space trajectory. Let us also note, that completeness of information is not assumed, since the environment is unlimited and is only partially visible. The criterion of optimality is the maximum information received at the current time. If the visibility changes, then the number of visible objects may also change, thus, the priority lattice and, hence, the path may change.

6. CONCLUSION

In this paper, we present the method of evaluating visual information about rewards along the path of a system (e.g., a robot), the method for recognizing objects around the system (robot), and the method for describing the process of studying the environment. We decompose the images of visible objects into sets of geometric primitives (geons) that the system's neural network is pre-trained to recognize. Complete combinations of these geons (geons' schemes) corresponding to unknown objects can be stored in the system's memory as new items. During the study of the object, the numbers of geons in such schemes increase, and they are accepted as the position rewards. Thus, the system should move in such a way as to place in its database as many geons associated with the object as possible. Later, we suppose to represent in detail a method for generating 3-D objects using geon schemes in order to train a neural network to recognize them without the database.

As one can see in Fig. 4.6 – 4.11, the calculation on the limited training database gives quite satisfactory geon recognition in images of real objects when the database includes not only automatically generated images, but also a number of hand-marked ones.

This approach may be useful when creating intelligent robotic systems. In these systems, "brains" can be transferred from one robot to another one. However, the first such system must go through the entire learning process as a real person, from childhood to adulthood. During the process, it must explore objects of the outside world as they are, recognize and classify them and evaluate their usefulness based on the goals of the system. Our approach offers one of the steps in this direction.

The method has obvious limitations: we only store information about the shape of objects. Therefore, the method is not suitable for recognizing lines and individual features of items. Only objects that can be decomposed into geometric primitives can be recognized.

Thus, we consider such a process of studying the environment by an intelligent system as some movement in the environment. We used the intuition of baby-like behaviour when exploring an object in the environment to simulate the behavior of a robot-like system. The movement of the system was represented as a game in the specific game category in which the environment corresponds to the Opponent. He provides the Proponent (system) with some information about the objects of the environment, which can also be considered as the goals of the system. Achievement of various goals is considered as parallel processes, which are presented as tensor products of the corresponding games, and form a comprehensive game.

Every game position has a certain reward (a geon set) that estimates the quantity of information provided by the environment. We demand the greatest total reward along the play of the system to choose the desired path. When the total reward of all parallel processes can no longer be improved, we can reduce the number of selected goals (processes for achieving the objects) to perhaps one at the end. An external method of the path calculation can modify this algorithm to create an objects' traversal path with the maximum cumulative reward.

Of all the possible, we choose those processes for achieving goals that have the highest estimations in the goal/object lattice of the system. It is so because every goal (an object of interest) and the corresponding process of achieving it have a definite correspondent truth value in the lattice. The higher the value lies on the lattice diagram, the higher the priority of the process. The partial order in the lattice is generated by attention degrees of the system to its goals.

Thus, we consider two types of lattice estimations: the goal lattice value determines the choice of the goal achieving processes from all possible ones, and the position rewards of the game (geon sets) determine the optimal path of these chosen processes in the environment. Let us note again that this path is the game play. It is not the real space trajectory. It only shows how to obtain the best position estimates, but there must be an external algorithm to build the trajectory in such a way as to get these assessments.

Such a model corresponds to the approach in which system intelligence is considered as the consequence of the mission or purpose of the system for practical use, i.e., the lattice of the goals of the system. In simple systems, e.g., ant colonies, such preference structures may be preexisting. In more complicated ones, these structures can be built and changed over the life of the system.

REFERENCES

1. Abramsky, S. & Jagadeesan, R. (1994) Games and full completeness for multiplicative linear logic, *J. of Symbolic Logic*, **59**(2), 543–574.
2. Abramsky, S., Jagadeesan, R. & Malacaria, P. (2000) Full abstraction for pcf, *Information and Computation*, **163**(2), 409–470.
3. Anshakov, O.N., Finn, V.K. & Vinogradov, D.V. (2008) Logical means for plausible reasoning of jsm-type, in: Finn V. (Ed.), *Multi-Valued Logics and Their Application*, **2**: *Logic in Artificial Intelligence Systems* Moscow, Russia: LKI, 226–236.
4. Biederman, I. (1987) Recognition-by-components: A theory of human image understanding, *Psychological Review* **94**(2), 115–147.
5. Birkhoff, G. (1967) *Lattice Theory* Rhode Island, Providence.
6. Cassimatis, N.L. (2012) Artificial intelligence and cognitive modeling have the same problem, in: *Theoretical Foundations of Artificial General Intelligence* Atlantis Press, 11–24.
7. Diane, S.A.K. (2018) Autonomous robots learning and social integration on the basis of modern cognitive technologies, *in Russian*, *Philosophy of Science and Technology* **23**(2), 89–102, doi:10.21146/2413-9084-2018-23-2-89-102.
8. Diane, S.A.K., Lesiv, E.A. & Zinchenko, I.V. (2019) Improving the efficiency of neural network models based on the synthesis of training examples in a virtual environment, *in Russian*, *Proc. 12-th Int. Conf. "Management of Large-Scale System Development" (MLSD'2019)*, Moscow, Russia, 288–294.
9. Diane, S.A.K. (2019) Dataset3d programme Url: https://github.com/Sekou/Dataset3D, Accessed 19 August 2021 .
10. Everitt, T. & Hutter, M. (2018) Universal artificial intelligence, in: Abbass H., Scholz, J. & Reid D. (Eds.), *Foundations of Trusted Autonom*y, Cham, Springer: Studies in Systems, Decision and Control,**117**, 15–46.
11. Fergus, R., Perona, P. & Zisserman, A. (2003) Object class recognition by unsupervised scale- invariant learning, *Proc. IEEE Conf. Comp. Vision and Patt. Recog.*, 264–271.
12. Felzenszwalb, P.F. & Huttenlocher, D.P. (2000) Efficient matching of pictorial structures, *Proc. IEEE Conf. Comp. Vision and Patt. Recog.*, **II**, 66–73.
13. Felzenszwalb, P.F. & Huttenlocher, D.P. (2004) Efficient graph-based image segmentation, *Int. J. of Comp. Vision*, **59** 167–181.
14. Felzenszwalb, P.F. & Huttenlocher, D.P. (2005) Pictorial structures for object recognition, *Int. J. of Comp. Vision*, **61**(1), 55–79.
15. Friedenberg, J. & Silverman, G. (2006) *Cognitive science: an introduction to the study of mind* Thousand Oaks – London – New Delhi, Sage Publications.
16. Girard, J.-Y. (1987) Linear logic, *Theoretical Computer Science* **50**, 1–102.
17. Hutter, M. (2005) *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*, Berlin, Springer.
18. Hutter, M. (2012) One decade of universal artificial intelligence, *Theoretical Foundations of Artificial General Intelligence* Atlantis Press, 66–88.
19. Hearn, D. & Baker, M.P. (2010) *Computer Graphics with OpenGL*, Prantice Hall.
20. Lafont, Y. (1999-2017) Linear logic pages, url: http://iml.univ-mrs.fr/lafont/pub/llpages.pdf, Accessed 19 August 2021.
21. Legovich, Y.S. & Maximov, D.Y. (2017) Selecting executor in a group of intellectual agents, *Automation and Remote Control*, **78**(7), 1341–1349.
22. Maximov, D.Y., et al, (2017) How the structure of system problems influences system behavior, *Automation and Remote Control*, **78**(4), 689–699.

23. Maximov, D.Y. (2016) Reconfiguring system hierarchies with multi–valued logic, *Automation and Remote Control*, **77**(3), 462–472.
24. Maximov, D. (2019) An optimal itinerary generation in a configuration space of large intel- lectual agent groups with linear logic, *Advances in Systems Science and Applications* **19**(4), 79–86, url: https://ijassa.ipu.ru/index.php/ijassa/article/view/829/513, Accessed 19 August 2021.
25. Maximov, D. (2018) Game semantics and linear logic in the cognition process, *arXiv.org*, **1812.11969**, 1–15.
26. Maximov, D. (2019) An optimal itinerary generation of large intellectual agent groups, *in Russian*, *Large System Management*, **78**, 46–70. doi:10.25728/ubs. 2019.78.3.
27. Maximov, D. (2020) Multi-valued neural networks and their use in decision making on the managment of a group of unmanned aerial vehicles *Proc. 13-th Int. Conf. "Management of Large-Scale System Development"(MLSD'2020)*, Moscow, Russia.
28. Mellies, P.-A. & Tabareau, N. (2010) Resource modalities in tensor logic, *Ann. Pure Appl. Logic*, **161**(5), 632–653.
29. Nuchter, A. (2013) An extension of the felzenszwalb-huttenlocher segmentation to 3d point clouds, *Proceedings of SPIE*, The International Society for Optical Engineering. doi:10.1117/12.2010527.
30. Solawetz, J. (2021) How to train a scaled-yolov4 object detection model Url: https://blog.paperspace.com/how-to-train-scaled-yolov4-object-detection/, Accessed 19 August 2021.
31. Solawetz, J. & Nelson, J. (2020) How to train yolov5 on a custom datasetUrl: https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/, Accessed 19 August 2021.
32. Sutton, R.S. & Barto, A.G.(2018) *Reinforcement learning: An introduction*, London, MIT Press.
33. Titsias, M.K. (2005) Unsupervised learning of multiple objects in images, *Phd thesis*, University of Edinburgh.
34. Weber, M., Welling, M. & Perona, P. (2000) Unsupervised learning of models for object recognition, *Proc. Fifth Europ. Conf. on Comp. Vision, ECCV 2000*, 18–32.
35. Welcome to albumentations documentation, url: https://albumentations.ai/docs/, Accessed 19 August 2021.
36. Zuffi, S., Freifeld, O. & Black, M.J. (2012) From pictorial structures to deformable structures, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Providence, IEEE, 3546–3553, doi:10.1109/CVPR.2012.6248098.
37. Zhu, X., Liu, Y., Qin, Z. & Li, J. (2017) Data augmentation in emotion classification using generative adversarial networks, arXiv.org, 1–15, Url: https://arxiv.org/pdf/1711.00648.pdf.