Visual Scenes Mining for Agent Awareness Module

Gang Ma[1,2], Zhentao Tang[2,3], Xi Yang[4], Zhongzhi Shi[1], and Kun Yang[5]

[1]The Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences. 100190 Beijing, China
{mag,shizz}@ics.ict.ac.cn
[2]University of Chinese Academy of Sciences. 100190 Beijing, China
[3]The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. 100190 Beijing, China
tangzhentao2016@ia.ac.cn
[4]Beijing Advanced Innovation Center For Future Education, Beijing Normal University. 100875 Beijing, China
xiyang85@bnu.edu.cn
[5]National Institute of Metrology. 100029 Beijing, China
yangkun@nim.ac.cn

Abstract. Most agents obtain knowledges from natural scenes through some single preestablished rules. In practice, those single rules can't achieve the aim to freely awareness the natural scenes, such as the visual scenes. Inspired by biological visual cortex (V1) and higher brain areas perceiving visual features, in this paper we propose an improved visual awareness module, called as visual scenes mining module, for the agent ABGP-CNN in order to directly mine the visual scenes information. Then ABGP-CNN with the visual scenes mining module is deployed on a toy car. The visual information mining from the nature scenes is served as the knowledges of the agent ABGP-CNN to drive the toy car. The toy car deployed the agent ABGP-CNN can easily understand the special natural visual scenes, and has the ability to plan its behaviors according to the visual information mining from the nature scenes. The application of the agent ABGP-CNN with visual scenes mining module enhances the capability of communication between the toy car and the natural visual scenes.

Keywords: Agent, Visual Scenes Mining, ABGP-CNN

1 Introduction

Rational agents have an explicit representation for their environment (sometimes called *world model*) and objectives they are trying to achieve. Rationality means that the agent will always perform the most promising actions (based on the knowledge about itself and sensed from the world) to achieve its objectives. For a rational agent faced with a complex natural scene, how to get knowledge from scenes to drive their actions? Most agent designer maybe have a common view that either create a virtual scene or set some single inflexible rules for agent to recognize surrounding.

There exist numerous agent architectures as mentioned above, such as BDI [1,16], AOP [19], SOAR [10] and 3APL [6]. In these architectures, the communication of

P. Perner (Ed.): ICDM 2017, LNAI 10357, pp. 166–180, 2017.
DOI: 10.1007/978-3-319-62701-4_13

information between agents and world is based on a single fixed rules as well. With respect to the theoretical foundation and the number of implemented and successfully applied systems, the Belief-Desire-Intention (BDI) architecture designed by Bratman [1] as a philosophical model for describing rational agents should be the most interesting and widespread agent architecture. Of course, there are also a wide range of agents characterized by the BDI architecture. Where one of these types is called ABGP (a 4-tuple $\langle Awareness, Belief, Goal, Plan\rangle$ BDI agent model shown in Figure 1) model proposed by Shi Z. [18].

ABGP model consists of the concepts of awareness, beliefs, goals, and plans. Awareness is an information pathway connecting to the world (including natural scenes and other agents in multi-agent system). Beliefs can be viewed as the agent's knowledge about its setting and itself. Goals make up the agent's wishes and drive the course of its actions. Plans represent agent's means to achieve its goals. However, ABGP model agent still has a disadvantage that just transfers a special fixed-format message from the world. Of course, there are many researchers built some usefull and interesting awareness modules, where one of works may be in [13] proposed ABGP-CNN model by introducing Convolution Neural Networks (CNN) as an environment visual pathway to implement the visual awareness module of the agent model ABGP. The ABGP-CNN model has the ability to directly capture the information from the natural scenes .

The literature [13] had described the functions and application fields of ABGP-CNN through a maze search experiment based on the agents implemented by ABGP-CNN model. However the experiment is carried out only through the software simulation on the computer, which means the experiment environment is very idealized. In practice there are many problems that we can't find out in the software simulation experiment, such that dynamically capturing visual information is affected by the light condition, how to find out a special interesting area from a large scene, etc.. Furthermore, ABGP-CNN as an intelligent model should be applied to the other fields with help of some hardware equipments, which can help human finish some more difficult tasks, such as disaster relief, danger detection, automatic drive, etc..

Motivated by above analysis, in this paper we will firstly improve the awareness module of ABGP-CNN agent, and then deploy the improved ABGP-CNN model on a true equipment. Where the improved ABGP-CNN will be served as a software driver (like human brain) to plan the behaviors of a toy car only through depending on the visual information that the toy car captures by its camera. The experiment can help us further explore the intelligent behaviors of the machine.

2 Agent Model ABGP-CNN

In computer science and artificial intelligence, agent can be viewed as perceiving its environment information through sensors and acting environment using effectors [17]. A agent model for rational agent should especially consider external perception and internal mental state. The external perception as a knowledge is created through interaction between an agent and its world. For the internal mental state, we can consult BDI model conceived by Bratman [1] as a theory of human practical reasoning. Reducing

the explanation framework for complex human behavior to the motivational stance is the especially attractive character [15].

An agent model ABGP-CNN (Figure 1) proposed by Ma G. [13] is one of the most typical agent model characterized by BDI architecture, it is represented as a 4-tuple framework as $\langle Awareness, Belief, Goal, Plan \rangle$. Where visual awareness implemented by Convolution Neural Networks (CNN) is an information pathway connecting to the world and a relationship between agents. Belief can be viewed as the agent's knowledge about its environment and itself. Goal represents the concrete motivations that influence an agent's behaviors. Plan is used to achieve agent's goals. Moreover, an important module, policy-driven reasoning in ABGP model, is used to handle a series of events to achieve plans selection.

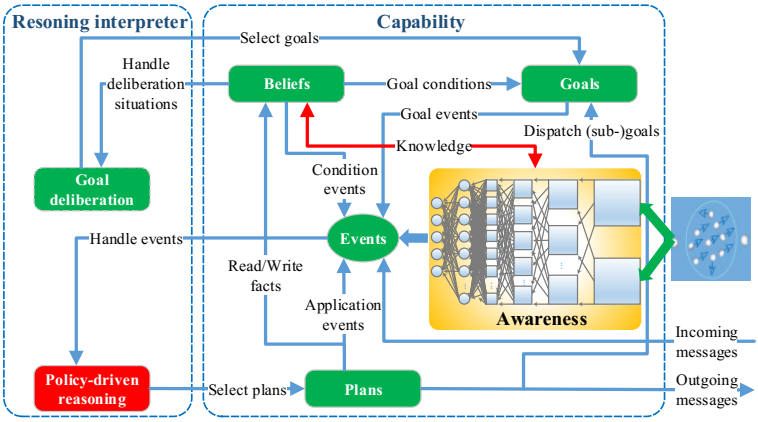

Fig. 1: Agent model ABGP-CNN [13].

For ABGP-CNN, the learning process of recognizing the natural scenes should mainly focus on how to train the CNN as its visual awareness module and how to build appropriate belief base, goals, and plans library. Training CNN includes what the multi-stages architecture is appropriate for the natural object recognition, what learning strategy is better. The aim of building beliefs, goals and plans is to achieve a series of agent's behaviors feedback according to the accepted environment information.

The implementation (Figure 2) of ABGP-CNN model adopts a declarative and a procedural approach to define its core components Awareness, Belief, Goal and Plan as well. The awareness and plan module have to be implemented as the ordinary Java classes that extend a certain framework class, thus providing a generic access to the BDI specific facilities. Belief and Goal module are specified in a so-called Agent Definition File (ADF) using an XML language (Figure 2). Within the XML agent definition files, any developer can use valid expressions to specify any designated properties. Some other information is also stored in ADF such as default arguments for launching the agent or service descriptions for registering the agent at a directory facilitator. Moreover, Awareness and Plan need to be declared in the ADF before they work.

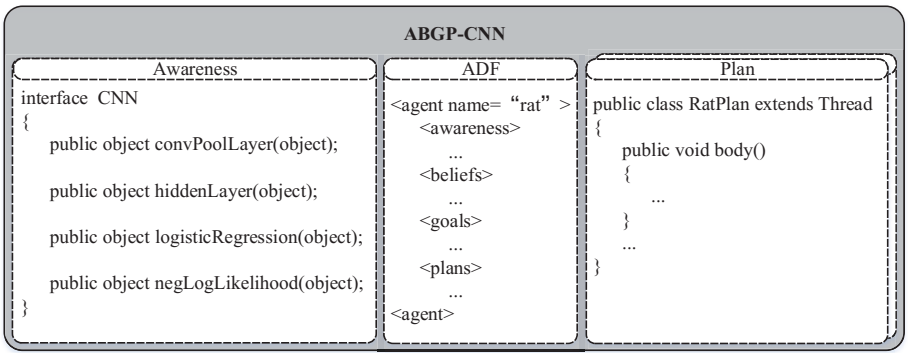

Fig. 2: Implementation of ABGP-CNN model[13].

Awareness is commonly viewed as an information path connecting to the environment. The ADF of ABGP-CNN provides a description of attributes for CNN, such as the number of CNN stages, the number of hidden layers, filter shape, pooling size, etc. Which can be any kinds of ordinary Java objects contained in the awareness set as an XML tuple. Those Java objects are stored as named facts.

Beliefs are some facts known by the agent about its environment and itself, which are usually defined in the ADF and accessed and modified from plans. Generally the facts can be described as an XML tuple with a name and a class through any kind of ordinary Java objects.

Plans in ABGP-CNN model can be considered as a series of concrete actions expecting to achieve some goals or tasks. ABGP-CNN model adopts the plan-library approach to represent the plans of an agent. Each plan contains a plan head defining the circumstances under which the plan may be selected and a plan body specifying the actions to be executed. In general for reusing plans in different agents, we need to decomposes concrete agent functionality into separate plan bodies, which are predefined courses of action implemented as Java classes.

Policy-driven Reasoning mainly make the plans (policies) selection by handling a series of events. A policy will directly or indirectly cause an action a_i to be taken, the result of which is that the system or component will make a deterministic or probabilistic transition to a new state S_i. Kephart et al. [14] outlined a unified framework for autonomously computing policies based upon the notions of states and actions. The agent policy model can be defined as a 4-tuple $P = \{S_t, A, S_g, U\}$, where S_t is the trigger state set at a given moment in time; A is the action set; S_g is the set of goal states; U is the goal state utility function set to assess the merits of the goal state level.

Algorithm 1 shows the functionality of those modules when they execute the interpret reasoning process. Goals deliberation constantly triggers awareness module to purposefully perceive visual information from the world the agent locates (extract visual information features $y : y_j = g_j \cdot tanh(\sum_i k_{i,j} * x_i), y \in D$) and convert those visual information features into the unified internal message events which are placed in the event queue (signal mapping $T : D \mapsto E$). According to goal events the event dispatcher continuously consumes the events from the event queue (corresponds to the

$OptionGenerator(EQ)$ function) and deliberates the events satisfying the goals (like $Deliberate(Options)$ function). Policy-driven reasoning module builds the applicable plan library for each selected event (similar to the $UpdatePlans(SelectedOptions)$ operation). In the $Execute(Plans)$ step Plan module in Figure 1 selects plans from the plan library and execute them by possibly utilizing meta-level reasoning facilities. Considering the competition among multiple plans, the user-defined meta-level reasoner will rank the plan candidates according to the priority of plans. The execution of plans is done stepwise, and directly drives agent's external and internal behavior. Each circulation of goal deliberation will be followed a so-called site-clearing work that means some successful or impossible plans will be dropped.

Algorithm 1: Interpret-Reasoning Process of ABGP-CNN

```
Initialize agent's states;
while not achieve goals do
    Deliberate goals;
    if world information INF or incoming messages IME was perceived then
        create internal event E according to INF or IME;
        fill internal event E into event queue EQ;
    end
    Options ← OptionGenerator(EQ);
    SelectedOptions ← Deliberate(Options);
    Plans ← UpdatePlans(SelectedOptions);
    Execute(Plans);
    Plans drive agent's behaviors;
    Drop successful or impossible plans;
end
```

3 Visual Scenes Mining Module

The experiment in [13] showed that CNN employed to construct the visual awareness module of ABGP behaves a high performance. Which means that the recognizing function of ABGP-CNN is mainly related to the visual awareness module. However the experiment is just carried out on the normative MNIST dataset in software simulation way. If CNN is directly used to construct the visual awareness module to recognize the handwrite digital in a complex visual scenes when ABGP-CNN is deployed on the toy car, it will behave a very low recognizing performance and can't achieve to correctly plan the behaviors of toy car. Therefore, we need to construct an improved visual awareness module of ABGP-CNN, called as the visual scenes mining module. In a complex visual scene, there are only some local elements containing interesting points which can guide the behaviors of toy car. We need to firslty locate the area with interesting points, then split it from the entire complex scene. At last the splited interesting area is recognized by CNN. Therefore some preliminaries, Pulse Coupled Neural Network,

Improved 2-Stage Sequential Similarity Detection Algorithm and Convolution Neural Networks, need to be introduced before building a novel visual scenes mining module.

3.1 Pulse Coupled Neural Network (PCNN)

The PCNN neuron in the standard PCNN model consists of three parts: the dendritic tree, the linking modulation, and the pulse generator, as shown in Figure 3.

The role of the dendritic tree is to receive the inputs from two kinds of receptive fields. Depending on the type of the receptive field, it is subdivided into two channels (the linking and the feeding). The linking receives local stimulus from the output of surrounding neurons, while the feeding, besides local stimulus, still receives external stimulus.

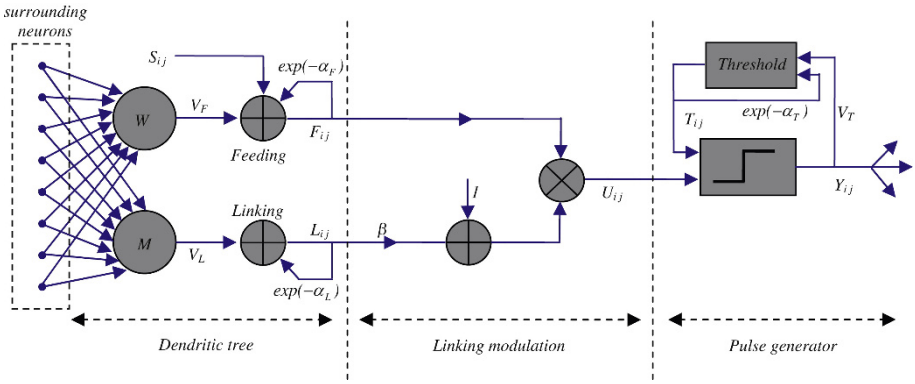

Fig. 3: Architecture of standard PCNN [21].

In the following expressions, the indexes i and j refer to the pixel location in the image, k and l refer to the dislocation in a symmetric neighborhood around one pixel, and n denotes the current iteration (discrete time step). Here n varies from 1 to N (N is the total number of iterations)

$$F_{ij}[n] = e^{-\alpha_F} F_{ij}[n-1] + V_F \sum_{k,l} w_{ijkl} Y_{ij}[n-1] + S_{ij}. \tag{1}$$

$$L_{ij}[n] = e^{-\alpha_L} L_{ij}[n-1] + V_L \sum_{k,l} m_{ijkl} Y_{ij}[n-1]. \tag{2}$$

$$U_{ij}[n] = F_{ij}[n](1 + \beta L_{ij}[n]). \tag{3}$$

$$Y_{ij}[n] = \begin{cases} 1, & U_{ij}[n] > T_{ij}[n]. \\ 0, & otherwise. \end{cases} \tag{4}$$

$$T_{ij}[n] = e^{-\alpha T} T_{ij}[n-1] + V_T Y_{ij}[n]. \tag{5}$$

The dendritic tree is given by Equation 1 and Equation 2. The two main components F and L are called feeding and linking, respectively. w_{ijkl} and m_{ijkl} are the synaptic weight coefficients and S is the external stimulus. V_F and V_L are normalizing constants. α_F and α_L are the time constants; generally, $\alpha_F < \alpha_L$. The linking modulation is given in Equation 3, where $U_{ij}[n]$ is the internal state of the neuron. β is the linking parameter and the pulse generator determines the firing events in the model in Equation 4. $Y_{ij}[n]$ depends on the internal state and threshold. The dynamic threshold of the neuron is Equation 5, where V_T and αT are normalized constant and time constant, respectively. Here is a brief review of the standard PCNN. The detailed description of the implementation of the standard PCNN model on digital computers can be found in literature [8]. More details about PCNN will be found in the literatures [11].

3.2 2-Stage of Sequential Similarity Detection Algorithm (2S-SSDA)

The basic idea of Sequential Similarity Detection Algorithm (SSDA) is that the algorithm constantly checks the similarity between realtime and reference images in each matching procedure. If the partial similarity between realtime and reference images is greatly different, the matching procedure will be terminated. Then the algorithm will choose another parameters to start a new matching procedure. However there are many disadvantages in SSDA, such that the large amount of computation exists in each matching procedure, many superfluous non-matching pixels need to be computed, and the algorithm needs to predefine a constant threshold T_k used to compute the error value in the matching procedure.

Theoretically, there are two ways to improve those issues. One is to reduce the searching space when the template T is used to search the reference image S. The other is to reduce the computational complexity of the similarity between the template T and the sub-image $S^{i,j}$. Next, we will give a 2-stage sequential similarity detection algorithm (2S-SSDA) [5] based above two theoretical inspires.

An image used to the matching operation needs to be preprocessed because of many superfluous information. Where we employ the preprocessing method that the normalized gray value is used as the reference image of probability density. Firstly the median filter method is used to equalize the image histogram. Then the edge features are extracted by Sobel edge detection operator. After extracting the edge features, we need to execute a sampling process on the edge features in order to improve the time efficiency of algorithm. The probability distribution of the processed image through above steps is described as:

$$P\{X = i, Y = j\} = \frac{H(i,j)}{\sum_{k=1}^{N} \sum_{s=1}^{N} H(k,s)} \tag{6}$$

in Equation 6 $H(i,j)$ indicates the gray value of the pixel (i,j) in the gray image. Then we sample a sequence $A(n)$ from the probability distribution of the gray image. n satisfies $n = |\{(i,j)\} H(i,j) > th|$, where th is a threshold. The roulette method can be

served to execute the sampling process. In the following the 2-stage of matching processing, rough matching and accurate matching, is executed to capture the best position of the matching sub-image.

In rough matching process, the candidate image set Q must be from the entire matched sub-image set S. Q has to satisfy the following conditions:

- The true matched point (i, j) must be in Q or Q-centered domain.
- In order to reduce the search space in the accurate matching, $|Q|$ should be as small as possible.
- Q should be easy and quick to get.

The distance between the sub-image and the template image is defined as $D(S^{i,j}, T)$:

$$D(S^{i,j}, T) = \sum_{k=1}^{n} d(S^{i,j}(A(k)), T(A(k))). \tag{7}$$

In the rough matching process, the elements contained in the set Q are determined by matching operation with the search step length h. In general, the candidate sub-image set Q should satisfy $Q = \{S^{i,j} | D(S^{i,j}, T) \leq \theta\}$, where θ is a threshold determined by the actual situation. A large Q will increase the matching time, while a small Q may lead to matching distortion, in most cases $|Q| = 1$.

In the whole process of rough matching, we don't make decision whether a sub-image is the target image when the matching process between the template image and the sub-image is finished, while Th is dynamically generated in the matching process. Where set $Th_1 = D(S^{1,1}, T)$, for the k-th matching, if s satisfies

$$\sum_{t=1}^{s} d(S^{i,j}(A(t)), T(A(t))) > Th_{k-1}; (s < n) \tag{8}$$

then $Th_k = Th_{k-1}$, and the current matching process is finished. Otherwise $Th_k = D(S^{i,j}, T)$.

In accurate matching process, the surface fitting method will be used to fit the 9 points of neighborhood area (Table 1) centered at the located pixel (i, j) generated from the rough matching process. Where the correlation coefficient function of image gray is defined as:

$$R(x, y) = \sum_{m=1}^{M} \sum_{n=1}^{M} S^{x,y}(m, n) \times T(m, n). \tag{9}$$

The correlation coefficient of the gray for 9 points can be computed through their actual locations.

From the above analysis we can see when all possible positions $R(x, y)$ achieve the maximum value, $D(S^{x,y})$ will obtain the minimum value, and the coordinate (x_0, y_0) of the point with the minimum value is the best matching location. So the quadratic surface fitting can be used to find the best matching location. Here the quadric surface is fitted by the least square method. Set the analytical formula of the quadratic surface:

$$f(x, y) = a_0 x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5. \tag{10}$$

Table 1: 9 points of neighborhood area of pixel (i, j).

$(i-1, j-1)$	$(i-1, j)$	$(i-1, j+1)$
$(i, j-1)$	(i, j)	$(i, j+1)$
$(i+1, j-1)$	$(i+1, j)$	$(i+1, j+1)$

The generated mean square error (MSE) between $f(x, y)$ and $R(x, y)$ is:

$$\delta(a_0 \sim a_5) = \sum_{i=1}^{9} [f(x_i, y_i) - R(x_i, y_i)]^2. \tag{11}$$

The stagnation point (x_0, y_0) of the quadric surface $f(x, y)$ can be obtained through computing the partial derivative of Equation 11. If $D(x_0, y_0) > 0$ and $\frac{\partial^2 f(x_0, y_0)}{\partial x^2} < 0$, (x_0, y_0) is the maximal value point [4]. If and only if there is one maximal value point for $\delta(a_0 \sim a_5)$ and $R(x, y)$ is a single value function, the location (x_0, y_0) is the best matching location when the sub-image $S^{x,y}$ and the template image are fully matched.

3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) consisted of multiple layers of small neuron collections has been adopted to recognize natural images [9]. In general, some local or global pooling layers may be included in Convolutional Neural Networks, which combine the outputs of neuron clusters [2]. They also consist of various combinations of convolutional layers and fully connected layers, with point-wise non-linearity applied at the end of or after each layer [3]. Generally, we call the combination with a filter bank layer, a non-linearity transformation layer, and a feature pooling layer, as a stage. Figure 4 shows a typical CNN framework composed of two stages.

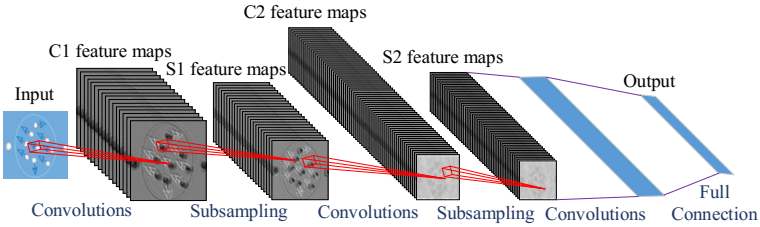

Fig. 4: A typical Convolution Neural Networks framework with two feature stages [13].

In filter bank layer, the input is a 3D array with n_1 2D feature maps of size $n_2 \times n_3$. Each component can be marked as $x_{i,j,k}$, and each feature map is denoted x_i. Where, The output is also a 3D array, and y consists of m_1 feature maps of size $m_2 \times m_3$. A trainable filter (so-called kernel) $k_{i,j}$ in the filter bank has size $l_1 \times l_2$ and connects input feature map x_i to output feature map y_j. The module computes: $y_j = b_j + \sum_i k_{ij} * x_i$, where $*$ is the 2D discrete convolution operator and b_j is a trainable bias parameter [7].

In non-linearity layer, a useful non-linearity function for natural image recognition is the rectified sigmoid $R_{abs} : abs(g_i \cdot tanh())$, where g_i is a trainable gain coefficient. The rectified sigmoid is sometimes followed by a subtractive and divisive local normalization N, which enforces a sort of local competition between adjacent features in a feature map, and between features at the same spatial location. The subtractive normalization operation for a given site $x_{i,j,k}$ computes: $v_{i,j,k} = x_{i,j,k} - \sum_{ipq} w_{pq} \cdot x_{i,j+p,k+q}$, where w_{pq} is a normalized truncated Gaussian weighting window (typically of size 9×9) normalized so that $\sum_{ipq} w_{pq} = 1$. The divisive normalization computes: $y_{i,j,k} = v_{i,j,k}/max(mean(\sigma_{j,k}), \sigma_{j,k})$, where $\sigma_{j,k} = (\sum_{ipq} w_{pq} \cdot v^2_{i,j+p,k+q})^{1/2}$ [12].

The purpose of feature pooling layer is to build robustness to small distortions, playing the same role as the complex cells in models of visual perception. P_A (*Average Pooling and Subsampling*) is a simplest way to compute the average values over a neighborhood in each feature map. The average operation is sometimes replaced by a P_M (*Max-Pooling*). Traditional CNN use a point-wise $tanh()$ after the pooling layer, but more recent models do not. Some CNNs dispense with the separate pooling layer entirely, but use strides larger than one in the filter bank layer to reduce the resolution [20] . In some recent versions of CNN, the pooling also pools similar feature at the same location, in addition to the same feature at nearby locations.

3.4 Architecture of Visual Scenes Mining Module

In general, the true environment that toy car runs on is a scenes with many complex elements. The interesting points guiding the behaviors of toy car always locate in some local elements of visual scenes. If we want to obtain the interesting information from these local elements. We firstly need to capture the location area of those elements with interesting points with help of location technologies (where PCNN and 2S-SSDA are adopted), then recognize the elements located in the interesting area through some recognition models (CNN is used here). In order to achieve the recognition task in the actual complex scenes, we proposed a novel visual scenes mining module (Figure 5) for ABGP-CNN based on PCNN, 2S-SSDA and CNN.

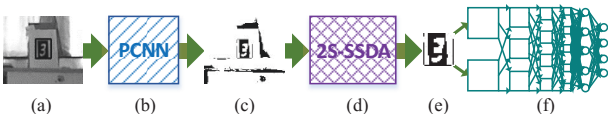

Fig. 5: Visual scenes mining module for the agent ABGP-CNN

Figure 5 illustrates the process that the visual scenes mining module captures and recognizes the interesting area in the actual complex scenes. In order to more explicitly show the function of the visual scenes mining module, the intermediate information products is shown in the architecture of visual scenes mining module. The visual scenes mining module consists of the modules (b), (d) and (f). The original image (a) with an actual complex scenes is processed by PCNN and output as the intermediate image (c).

Then the interesting area (e) is captured by 2S-SSDA from the intermediate image (c). At last CNN (f) will further recognize the interesting area and send the recognized information to the event module of ABGP-CNN, which helps ABGP-CNN make a series of activity plans.

In this paper, the handwrite digital with the value $0 \sim 3$ from MNIST data set is still used as the guidepost to guide the behaviors of toy car, because there are only four types of activities for toy car, namely '0' denotes moving on, '1' moving back, '2' turning left, '3' turning right. Next, we will give some preliminaries about PCNN, 2S-SSDA and CNN in order to have a better description about visual scenes mining module in ABGP-CNN in the following.

4 Application to Toy Car

The application of ABGP-CNN model in the literature [13] was mainly carried out through software simulation about the agent implemented by ABGP-CNN. For further exploring the practical application of ABGP-CNN, here we will deploy the ABGP-CNN model on a self-made toy car as a software control system like human brain. The toy car with ABGP-CNN model can freely plan its behaviors according to its belief, goal, plan and the visual information mined by the visual scenes mining module. The detialed information flow is shown in Figure 6.

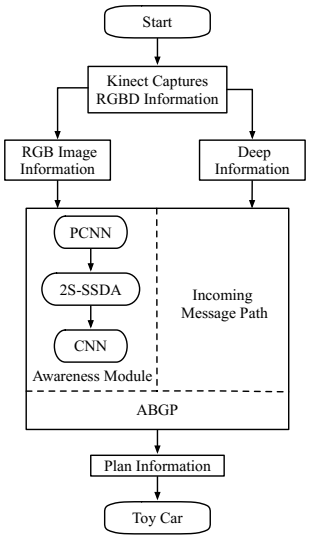

Fig. 6: Information Flow of Toy Car with ABGP Model

The self-made toy car consists of Kinect camera, STM32 control module, wireless receiving module, motor drive module, serial communication module, car body struc-

ture and a central controller implemented by a computer running ABGP-CNN model. The architecture of toy car is illustrated in Figure 7.

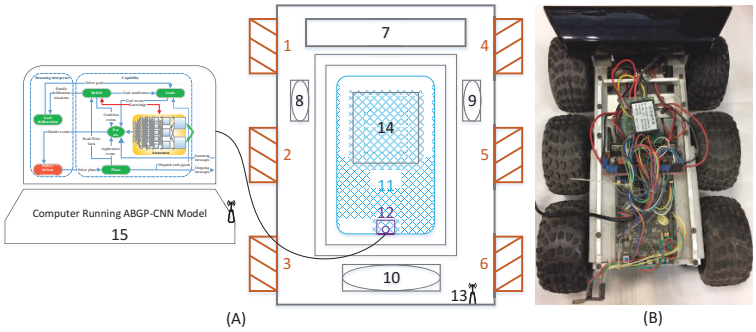

Fig. 7: (A) Architecture of toy car. 1∼6: Direct current (DC) motors. 7: Visual information collector. 8: Driver module of DC motors 1 and 2. 9: Driver module of DC motors 4 and 5. 10: Driver module of DC motors 3 and 6. 11: Circuit board. 12: Serial communication interface (SCI). 13: Wireless receiving device. 14: Main control module. 15: Computer running ABGP-CNN Model. (B) Actual toy car.

All modules in the architecture of toy car work collaboratively. Visual information collecting depends on Kinect with the RGB and deep camera. The RGB-D (RGB and deep) information from Kinect will be sent to the computer running ABGP-CNN model through serial communication interface (The visual information is transported in wire way because of large amounts of data). The visual scenes mining module in ABGP-CNN further recognizes the visual information from SCI, and a series of behaviors of the toy car will be planned immediately based on beliefs, goals, and the recognized visual information from the visual scenes mining module. The behaviors as a command controlling the toy car will be sent from the *Outgoing Message* interface in ABGP-CNN in radio wave way, then the wireless receiving device in the architecture of toy car will receive and transport the command to the main control module.

In the architecture of toy car, the serial communication interface (SCI) is implemented by RS-232, which provide the real-time wired communication between the main control module and the computer running ABGP-CNN model. STM32F103ZET6 is employed to construct the main control module, which receives and processes the control signals from SCI RS-232. The processed control signals will be served to control the speed and steering of the DC motors in Pulse-Width Modulation (PWM) square wave way, which achieves to control the speed and direction of toy car.

5 Experiment Analysis

In the experiment that ABGP-CNN is deployed on a toy car, the toy car is designed to have 4 basic behaviors moving on, moving back, turning left and turning right. There-

fore, we construct a sub-MNIST dataset, called *mnist0_3*, extracting 4 types of handwritten digits with flag '0' denoting moving on, '1' moving back, '2' turning left and '3' turning right from the original MNIST dataset. The dataset *mnist0_3* consists of 20679 training samples, 4075 validating samples, 4257 testing samples.

When the ABGP-CNN agent with visual scenes mining module deployed on the toy car recognizes the nature scenes, the RGB-D camera of Kinect in the toy car will collect in real time the images in the actual scenes. Each image from RGB-D camera will be converted to a gray image, and PCNN is employed to search a interesting area (handwrite digit area) on the gray image. Once the interesting area is found, it will be split out from the gray image and binarized as a binary image. At last the binarized image is linearly compressed to a smaller image with size of 28×28, because the training image of CNN is from a subset of MNIST dataset in which each image is the size of 28×28 as well.

The visual scenes mining module of ABGP-CNN is mainly implemented through Open NI, Open CV and Theano software environment.

- Open NI provides the function interface for extracting the Kinect RGB-D image, namely the depth information of each pixel. Which mainly tells us the distance between the toy car and the target object.
- Open CV mainly provides some function interfaces for generating gray images and template matching methods.
- Theano is served to implement CNN.

In order to validate that the visual scenes mining module can work correctly when ABGP-CNN is deployed on toy car, the experiment is carried out on three different light environments (Table 2), incandescent lamp light, fluorescent lamp light and natural light. In those three different light conditions, we design an actual road sand table (the architecture is same as [13]) with 32 different four types of guideposts (handwritten digits '0' denoting moving on, '1' moving back, '2' turning left and '3' turning right) from *mnist0_3* dataset. The toy car with ABGP-CNN model running on the road sand table can correctly plan its behaviors through depending on the visual scenes mining module recognizes the guideposts in the path.

Table 2: Recognition rate of ABGP-CNN for 1000 tests. ABGP-CNN with the visual scenes mining module deployed on toy car on condition of incandescent lamp light (ILL), fluorescent lamp light (FLL) and natural light (NL). Original ABGP-CNN deployed on toy car (OATC) and software simulation platform(OASP).

Experiment Type	ILL	FLL	NL	OATC	OASP
Recognition Rate	73.25%	83.47%	85.60%	52.30%	99.73%

From the experimental results in Table 2 can be seen, the recognition rate has a great fluctuation under the different light conditions, because the optical camera is sensitive to light conditions. The closer the light is to natural light, the lower the optical camera is interfered, and the higher the recognition rate. The experiment of the original ABGP-CNN deployed on the software simulation platform shows that the recognition rate of

ABGP-CNN has reached the excellent performance of 99.73%. While the recognition rate of the original ABGP-CNN directly deployed on toy car drops to the low performance of 52.30%. However the recognition rate of the ABGP-CNN with visual scenes mining module keep a high performance of 85.60% when it is deployed on an actual toy car on the condition of natural light. Though the ABGP-CNN with visual scenes mining module can achieve a high performance, we still have the following conclusions why it can't reach a higher performance.

- External environment. The sharpness of guideposts and the optical characteristics of camera result to the deviation of collecting images.
- Image processing. The binarized threshold setting is affected by the changes of the external environment. The image compressing in the two-dimensional plane may produces the linear compression error. The above two processes may decrease the recognition rate.
- CNN training process. The training samples of CNN is from the *mins0_3*, while the samples used to recognition test is collected in real time from the external environment. Because we lack of the training samples collecting in real time from the actual external environment.

If we want to get the better recognition rate for ABGP-CNN in the actual application, the ABGP-CNN with the visual scenes mining module can be optimized further based on the above three reasons.

6 Conclusion and Future Work

The agent built by ABGP-CNN can plan its behaviors through the environment visual information. However all applications for ABGP-CNN model are carried out only through the software simulation. In this paper we deploy ABGP-CNN on an actual toy car and improve the visual scenes mining module, such that ABGP-CNN can directly mine the visual information from the true natural scenes. ABGP-CNN with the novel visual scenes mining module can directly conduct the behaviors of toy car, which enhances the capability of communication between the toy car and true environment, and improves the intelligence of toy car. The current visual scenes mining module does not still satisfy high accuracy behaviors plan of toy car. For future work, we will focus on improving awareness accuracy of visual scenes mining module in true environment.

Acknowledgments

This work is supported by the National Basic Research Program of China (973) (No. 2013CB329502), the National Natural Science Foundation of China (No. 61035003, 61202212, 61072085), the National Key Technology Research and Development Program (No. 2014BAK02B07).

References

1. Bratman, M.E.: Intention, Plans, and Practical Reason. Cambridge University Press (1987)
2. Ciresan, D.C., Meier, U., Masci, J., Maria Gambardella, L., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: Proceedings of the 21th International Joint Conference on Artificial Intelligence. vol. 22, pp. 1237–1242 (2011)
3. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Proceedings of the 25th Conference on Computer Vision an Pattern Recognition. pp. 3642–3649. IEEE (2012)
4. Department of Applied Mathematics in Tongji University: Advanced Mathematics. Higher Education Press, Beijing (2014)
5. Duan, X.: Improvement of Matching Algorithm Based on Gray Image. Master's thesis, Central South University (2012)
6. Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.J.C.: Agent programming in 3apl. Journal of Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
7. Jarrett, K., Kavukcuoglu, K., Lecun, Y.: What is the best multi-stage architecture for object recognition? In: IEEE 12th International Conference on Computer Vision. pp. 2146–2153 (2009)
8. Johnson, J.L., Padgett, M.L.: Pcnn models and applications. IEEE Transactions on Neural Networks 10(3), 480–498 (May 1999)
9. Korekado, K., Morie, T., Nomura, O., Ando, H., Matsugu, M., Iwata, A.: A convolutional neural network vlsi for image recognition using merged/mixed analog-digital architecture. Knowledge-Based Intelligent Information and Engineering Systems pp. 169–176 (2003)
10. Lehman, J.F., Laird, J., Rosenbloom, P.: A gentle introduction to soar, an architecture for human cognition. Invitation to Cognitive Science 4, 212–249 (1996)
11. Lindblad, T., Kinser, J.: Image Processing Using Pulse-Coupled Neural Networks. Springer-Verlag Berlin Heidelberg (2005)
12. Lyu, S., Simoncelli, E.P., Hughes, H.: Nonlinear image representation using divisive normalization. In: Proceedings of the 21th Conference on Computer Vision an Pattern Recognition. pp. 23–28 (2008)
13. Ma, G., Yang, X., Lu, C., Zhang, B., Shi, Z.: A visual awareness pathway in cognitive model abgp. High Technology Letters 22(41), 395–403 (2016)
14. O., K.J., E., W.W.: An artificial intelligence perspective on autonomic computing policies. In: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks. pp. 3–12 (2004)
15. Pokahr, A., Braubach, L.: The active components approach for distributed systems development. International Journal of Parallel, Emergent and Distributed Systems 28(4), 321–369 (2013)
16. Rao, A.S., Georgeff, M.P.: Bdi agents: From theory to practice. In: Proceedings of the First International Conference on Multi-Agent Systems. pp. 312–319 (1995)
17. Shi, Z., Wang, X., Yue, J.: Cognitive cycle in mind model cam. International Journal of In-telligence Science 1(2), 25–34 (2011)
18. Shi, Z., Zhang, J., Yue, J., Yang, X.: A cognitive model for multi-agent collaboration. International Journal of Intelligence Science 4(1), 1–6 (2013)
19. Shoham, Y.: Agent-oriented programming. Artificial Intelligence 60(1), 51–92 (1993)
20. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: the 12th International Conference on Document Analysis and Recognition. pp. 958–963 (2003)
21. Wang, Z., Ma, Y., Gu, J.: Multi-focus image fusion using pcnn. Pattern Recognition 43(6), 2003 – 2016 (2010)