International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.4 (2015), pp.183-192
http://dx.doi.org/10.14257/ijmue.2015.10.4.18

An Approach to Robot Commonsense Reasoning

Kejun Wang, Tongchun Du and Xiaofei Yang

College of Automation, Harbin Engineering University
wangkejun@hrbeu.edu.cn, {tongchundu, xiaofeiy408}@gmail.com

Abstract

To perform missions assigned by people, the mobile robot needs to learn both itself and the world it is in, to use the learned knowledge to reason and make decisions to resolve problems, and to guide further learning, and to establish its knowledge base which contains huge amounts of common sense. During the process of executing missions by reasoning and learning, it is critical to enable domain-specific robots be more tolerant rather than puzzled when encountering perturbations. In this paper, active logic is improved to deal with contradicted beliefs, metacognitive loop is incorporated to supervise and guide the whole reasoning process. Finally, route crack experiments state that this metacognitive loop and improved active logic based method can handle robot commonsense reasoning in an efficient way and be comparatively robust with perturbations.

Keywords: *commonsense reasoning, robot, metacognitive loop, improved active logic*

1. Introduction

One long-standing dream of researchers in artificial intelligence and cognition science communities is to construct a flexible and human-level intelligent system/agent, such as a so-called autonomous and/or intelligent robot [1]. To illustrate what "flexible" means, let us consider some contrasted examples, some domain or mission specific robots would be disordered, have no idea about what to do, even be injured, and finally result in failure, when the environment deviates a little from their expectations. For example, an unmanned vehicle can automatically avoid obstacles and select flat routes to go through, but suppose there occurs a big ditch on its frontal route (note that this is different from a traffic jam), then how should the unmanned vehicle do to handle this situation, to save itself, and to lead to a successful mission executing? Another example is the rescue robot which usually uses infrared sensors to look for survivals at night, but if a building is on fire and thus makes infrared sensors unsuitable to detect, how can the rescue robot find this changed circumstance and adjust both its sensors and detecting algorithms automatically? In this paper, "perturbation" is defined as all situations that are different from what are pre-designed, *e.g.*, sensor failure, environment change, etc. When encountering a perturbation, a robot can protect itself, recover from the perturbation, re-assign its goal, and change policies to guide executing missions. The abilities in this case are to some degree what we intend to think of as "flexible" [1].

As one vehicle of Artificial Intelligent, a mobile robot often situates in complex and dynamic environments. To perform missions assigned by people, the mobile robot needs to learn both itself and the world it is in [5, 9], to use the learned knowledge to reason and make decisions to resolve problems and to guide further learning, and to establish its knowledge base which contains huge amounts of common sense [3]. Similar to but a little different from an expert system which also needs huge amounts of expertise, what a robot needs is a lot of common sense, *e.g.*, a household robot needs to learn about a kitchen's environment including how a door looks like, what a plugin is, and how to plan a path from one location to another, etc. Such kinds of common sense can be initially added by the agent designer to the knowledge base (KB) or be obtained through learning, perception, and reasoning. Reasoning rules of course play one critical role in KB. As is concluded by many researchers, the main challenges of commonsense reasoning include but not limit to that common sense cannot be fully described, that the expressions are imprecise and incompatible, and that reasoning is non-monotonic. In the past decades, non-monotonic reasoning always played the core role in commonsense reasoning, several derived algorithms also paid much attention to non-monotonic problems and tried to extend KB of commonsense to contain as many kinds of perturbations as possible. When there exist contradicted beliefs (facts and rules), some methods even simply ignore those contradictions and use the remaining knowledge to reason. It seems that contradictions and perturbations are so horrific that everyone want to avoid them. However, in our opinion, much of human knowledge are just gained from mistakes and contradictions, so even contradicted beliefs could help commonsense reasoning. What is more, it will be a labor intensive job for a reasoning machine designer to consider all kinds of perturbations and put them in KB. Along this cue, we argue it is critical to design a domain general perturbation handling mechanism to supervise the commonsense reasoning process. In the paper, metacognitive loop is just playing this key role.

KB is continuously updated as the reasoning process goes. How to let the reasoning machine not only reason about rules and knowledge but also make notes of the reasoning process, so that when contradicted beliefs occur or some logic errors appear, the system can conveniently consult and find out the problems. The answer is active logic which is time situated, can not only reason in real time but also reason about time, and is able to detect contradicted beliefs efficiently. We improve its contradiction handling ability by adding some rules and definitions.

To supply the context of our idea, a “route crash” example will be presented in Section II; Section III introduces the metacognitive loop (MCL); the principles of active logic lie in Section IV; we give some details about how to realize our idea about route crash example in Section V, also you will see our current plans and future research directions there.

2. Route Crash Example

Robot Jack moves at a normal speed from location A to location B along a planned route AB. At location C on the midway, his sensors find that a crash occurs which may block the way to B. How should Jack react?

Let’s stop to make some explanations. Jack has its KB which contains some knowledge about a crash, *e.g.*, he can cross those crashes with finite width by accelerating. Since the crash is not included in the route plan, Jack thinks of it as a perturbation to going through the route AB. Herein, the original belief (*e.g.*, *Unblocked* (*AB*)) is not suitable to describe the current world.

Suppose there is a pre-equipped crash perturbation handling method with Jack. It should be like this: Jack will observe the crash’s width and decide whether it is becoming larger, and observe route *AC* to see if it is the same as before and if it is still safe (that is Jack’s initial expectations: no perturbation and safety). If he is sure that the crash is not too wide and not becoming wider, that route *CB* is still safe, and that according to his knowledge he can cross the crash by accelerating, then he will accelerate to cross the crash. If Jack finds that the crash is too wide to allow him cross or the crash is becoming wider and wider, and route *AC* is still safe, then he would probably choose to accelerate to return to location *A*. It is important to note that “accelerate to cross” and “accelerate to return” are much different from “move normally”.

In this route crash case, Jack’s actions involve several kinds of abilities, he needs to be able to firstly plan the path AB, to have KB including many common sense about a crash, some expectations about going through the route and how the route looks like, to be able to observe the width and the state of the crash and compare the value to his experience, to evaluate the danger level to decide to go forth or return, and to possibly learn how to accelerate. Reasoning, memory, vision, and learning all are required to bring Jack closer to the so-called intelligent agent [4]. But the core role in this case is obviously the perturbation detecting and handling mechanism/framework. Our long term research aims to put some state-of-the-art technologies in this agent and has gained some successes. Despite there is still an unforeseen way to go, we are pleased to present some parts of our work here.

3. Metacognitive Loop

Metacognitive loop (MCL) is composed of setting expectations, comparing observations to expectations, evaluating and explaining the perturbations, reacting to the perturbations to guide the agent, and adjusting the expectations [2, 4, 11].

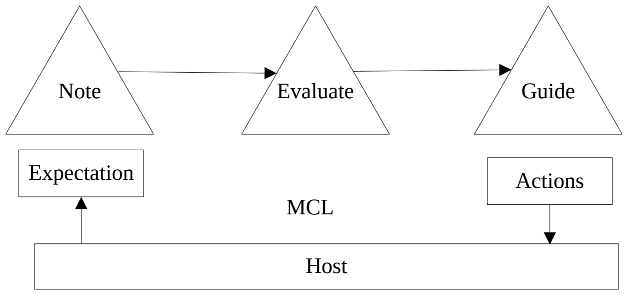


Figure 1. Perturbation Handling Procedures of MCL [2]

The contribution of MCL to the route crash example is detailed as below. Jack has an expectation about route *AB* (*i.e.* unblocked), when there is a discrepancy between the observation and this expectation, Jack views the observation as a perturbation; evaluates the perturbation according to his available knowledge; then supplies corresponding actions to guide executing subsequent missions, e. g, accelerate to cross the crash and go along route *CB* or accelerate to return to *A*; and finally adjusts the original expectations such as *MoveTo*(*A*), and adds these two pieces of knowledge, that accelerating to cross or accelerating to return can deal with route crashes, to KB.

We could say with equipping MCL as the perturbation handling framework, if Jack owns a lot of common sense about crashes, he can do well in dealing with crash perturbations. Moreover, even though Jack knows little about crashes, MCL has a suit of simple but efficient methods to resolve anomalies, *e.g.*, evaluation methodology evaluates that the perturbation may do harm to Jack but cannot get detailed information about the crash, then guidance methodology just selects an avoiding policy to escape from this perturbation. If Jack is sure that the crash is a perturbation but has no idea how to react, he will ask for help or just terminate the mission. In practice, MCL aims to enable the system more tolerant to perturbations rather than limit the scope of perturbations.

There are three key issues with MCL. First, how to select detecting technologies according to the world's changes? Second, how to fast and reasonably evaluate the perturbation's level? That is another reason why MCL is worthy. If the level of perturbations is low and cannot affect the mission executing, *e.g.*, the crash is not wide and allows Jack to cross, then Jack (the host) should not change expectations, because in this case accelerating to cross the crash may be at a high cost. Third, what kind of policy is most efficient to lead react corresponding to some specific perturbations? These policies at the least include go-back policy, which means if knowledge about the perturbation is very limited, then the agent should simply return to the starting location and learn from scratch, *e.g.*, if receiving a recommended action “accelerate to cross” but he does not know how to accelerate, Jack then needs to learn the new notation.

MCL can not only monitor system’s performance, but also know its descriptions about perturbation are not suitable, and then employ methods such as learning and updating beliefs to improve its performance. Therefore, different kinds of descriptions, reasoning methods, and self-monitoring should be combined and cooperate in an efficient way. Besides, as different domains need different kinds of methodologies and rules, the designer of MCL should guarantee the knowledge and rules are suitable to the specific domain. The system should be aware of the world shifting in order to adjust itself, which requires abilities of high-level self-awareness and self-control. Self-learning and self-improving are too in need to keep the system adaptive from one domain to another by learning and choosing suitable methodologies. Above are the salient characteristics of system with MCL [1, 2, 4, 7].

4. Active Logic

KB consists of linguistic descriptions from users, environment descriptions, domain states, and rules added by managers. It can also divided into rules, facts, and states, all are called knowledge [1, 2, 6]. The process that active logic reasons about knowledge and makes notes of the process is depicted in Figure 2 [2].

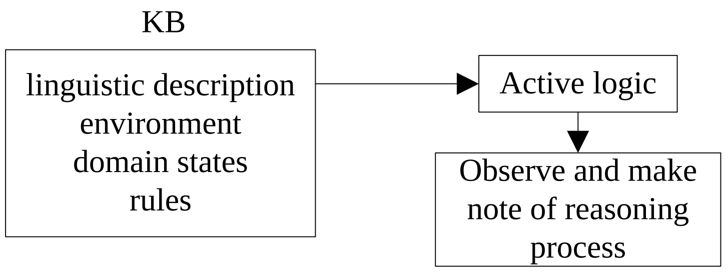


Figure 2. Procedures of Reasoning about Knowledge in KB of Active Logic

Active logic is time situated, which can reason in real time as well as reason about time. All rules and knowledge are labeled with timestamps. The symbolic representations are listed here. $\wedge$ means logic and, $\rightarrow$ induct to, $\neg$ not, $\in$ contained by, and $\|$ irrelevant. Some reasoning rules of active logic are show here [10].

(1) “Now” awareness. All rules and knowledge are coupled with timestamps, e. g, the current step number is i, then next time step is i+1.

$$i\text{: } Now(i)$$
$$i+1\text{: } Now(i+1)$$

(2) Modus Ponens. If at time step i KB has knowledge A and the rule B can be obtained with A, then next step, B is gained and added to KB. There is also extended modus ponens.

$$\frac{i\text{: } \ldots, \propto, (\propto \rightarrow \beta)}{i+1\text{: } \ldots,\ \beta} \qquad Modus\ Ponens$$

$$\frac{i\text{: } \ldots, P_1 a, \ldots, P_n a, \forall x[(P_1 x \wedge \ldots \wedge P_n x) \rightarrow Qx]}{i+1\text{: } \ldots, Qa}$$

Extended Modus Ponens

(3) Inheritance. Knowledge at the former time step will be inherited at the next time step.

$$\frac{i: \ldots, \alpha}{i+1: \ldots, \alpha} \quad Inheritance$$

(4) Modus Tollens. If the conclusions do not exist, then corresponding preconditions are not fulfilled.

$$\frac{i: \ldots, !\beta, (\alpha \rightarrow \beta)}{i+1: \ldots, !\alpha} \quad Modus\ Tollens$$

(5) Contradiction detection. If novel knowledge (derived by reasoning or obtained through observation) in KB is conflicted with old knowledge, active logic can detect this conflict and delete the unsuitable knowledge to keep KB consistent. In active logic, the contradiction detection and handling are completed within two time steps.

$$\frac{i: \ldots, \propto, ! \propto}{i+1: \ldots, contra(\alpha, !\alpha)} \quad Contradiction\ detection$$

It is easy to detect contradictions but hard to resolve them. In many cases, we cannot just delete contradictions, nor could we ignore them but only use consistent knowledge to reason. This paper makes some revisions to improve the contradiction detection and handling abilities of active logic.

(6) We define Fact1 and Fact2 as two facts, and CommonSense1 and CommonSense2 as two common sense. Facts are concrete and special, while common sense are abstract and general. The relations between facts and common sense include contained, negated, and irrelevant.

Fact1 ‖ *Fact2* indicates *Fact1* is irrelevant to *Fact2*.
Fact1 = ¬ *Fact2* indicates *Fact1* is the negated form of *Fact2*.
Fact1 ∈ *Fact2* means *Fact1* is contained by *Fact2*.

Contra (*CommonSense1*, *CommonSense2*) indicates that *CommonSense1* and *CommonSense2* are a pair of contradicted knowledge.

If we have

Fact1 → *CommonSense1*
Fact2 → *CommonSense2*
Contra (CommonSense1, CommonSense2), and
Fact1 ∈ *Fact2*,

Then keep *CommonSense1* and delete *CommonSense2*, because *Fact1* is a special case of *Fact2*, we call this fact first policy. But if *Fact1* ‖ *Fact2* which means two irrelevant facts induct two contradicted conclusions, then delete both two conclusions or just ignore them waiting for more knowledge to process later.

Besides, the contained relations of facts have transitivity, that is to say, if *Fact1* ∈ *Fact2*, *Fact2* ∈ *Fact3*, then we get *Fact1* ∈ *Fact3*.

(7) For irrelevant facts and common sense, if they are not contradicted with preconditions of a rule, they do not influence the reasoning.

If *Fact1* ‖ *Fact2*
Fact1 → *CommonSense1*, then
Fact1 ∧ *Fact2* → *CommonSense1*.

(8) Modus Ponens does not have transitivity. If $A \rightarrow B$, $B \rightarrow C$, now given A, we could get $A \wedge (A \rightarrow B) \wedge (B \rightarrow C) \rightarrow C$ rather than $A \rightarrow C$. Similarly, Modus Tollens does not have transitivity.

We would like to test the improved active logic using some benchmark examples of commonsense reasoning below.

(a) If it is rainy, then the floor is wet. It is rainy, so the floor is wet.

$$Rainy(Weather) \rightarrow Wet(Floor)$$

(b) Birds can fly, wiki is a bird, so wiki can fly.

$$Birds(wiki) \wedge ((Birds(X) \rightarrow CanFly(X)) \rightarrow CanFly(wiki)$$

(c) Birds can fly, wiki is a bird, but wiki cannot fly, so wiki cannot fly.

$$Birds(wiki) \wedge (Birds(X) \rightarrow CanFly(X)) \wedge (\neg CanFly(wiki)) \wedge (wiki \in Birds)$$
$$\rightarrow \neg(CanFly(wiki))$$

(d) If it is rainy, then the floor is wet. It is rainy and windy, so the floor is wet.

$$Windy(Weather) \wedge (Rainy(Weather) \rightarrow Wet(Floor)) \wedge (Windy(Weather) \| Rainy(Weather))$$
$$\rightarrow Wet(Floor)$$

(e) Nixon Christians are pacifists, communists are not pacifists, Nixon is a Christian, and Nixon is a communist, so Nixon is (pacifist or not)?

$$(Christian(X) \rightarrow Pacifist(\mathrm{X})) \wedge (Communist(X) \rightarrow \neg Pacifist(X))$$
$$\wedge Christian(Nixon) \wedge Communist(Nixon) \wedge (Christian(X) \| Communist(X))$$
$$\rightarrow Contra(Pacifist(Nixon), \neg Pacifist(Nixon))$$

(f) Penguin is a bird, birds can fly, penguin cannot fly, wiki is a bird, and wiki is a penguin, so wiki (can fly or not)?

$$\mathrm{Birds(wiki)} \wedge (Birds(X) \rightarrow CanFly(X)) \wedge (Penguin(X) \rightarrow (\neg CanFly(X)))$$
$$\wedge (Penguin(\mathrm{wiki})) \wedge Penguin(X) \in Birds(X)$$
$$\rightarrow \neg(CanFly(wiki))$$

(g) If Carter died in 1979, then he would not fail in the selection in 1980. If Carter did not fail in the selection in 1980, then Reagan would not be selected as the president. So if Carter died in 1979, Reagan would not be selected as the president. As is talked before, Modus Ponens does not have transitivity, so this inference is not set up.

$$Died(Carter) \rightarrow \neg FailedSel(Carter) \wedge \neg FailedSel(\mathrm{Carter}) \rightarrow FailedSel(Reagan)$$
$$\rightarrow (Died(Carter) \wedge (\neg FailedSel(Carter) \rightarrow FailedSel(Reagan))$$

Supplemented with rules (6), (7), and (8), active logic get more suitable for commonsense reasoning. Besides every piece of conclusion, precondition, reasoning rule, and every step of reasoning process are labeled with timestamps, the technology to handle contradictions can be summarized as that reasoning rules do not have the transitivity, that the conclusions derived by specific preconditions with small extension are superior to those derived by general preconditions with large extension, and that those contradicted conclusions derived by irrelevant preconditions should be both deleted or pended waiting for sufficient knowledge to handle.

5. Experiments and Analysis

The route crash example involves three basic problems in commonsense reasoning: (1) Beliefs would change as the time goes, that is, Jack expected the route is unblocked and then found there was a crash blocking his route. (2) If the agent does not have knowledge about crashes, such as how to observe the changes of crashes, then it could not describe and make decisions properly. That is, knowledge in KB could not match the environment. (3) Accelerating to cross or return are different from moving normally, so the original expectations need to be updated.

Fortunately, those characteristics of active logic are designed to deal with these problems, which include reasoning step by step, labeling rules and knowledge with timestamps, technologies to resolve contradictions, and updating expectations and knowledge. Some details about the route crash example are shown below.

Initializing KB and expectations. *UnBlocked*(*AB*), *t*=0. *LocateAt*(*A*), *t*=0. *MoveNormal*(*AB*).
ArriveAt(*B*). *SpectFrontRoute*. *SpectBackRoute*.

Set some kinds of perturbations. ¬*UnBlocked*(*AB*). *CrashAt*(*X*), *X*∈*AB*. *FireAt*(*X*), *X*∈ *AB* indicates there is fire on route *AB*. *BloodAt*(*X*), *X*∈*AB* indicates there is flood on route *AB*.

Evaluating the perturbations:

CrashAt(*X*)∧*CrashWidth*(*Width*)∧*Width*>5→*Danger*(*Crash*).
CrashAt(*X*)∧*CrashWidth*(*Width*)∧*Width*<5∧¬*CrashGrowing*(*X*) → *Safe*(*Crash*).
CrashAt(*X*)∧*CrashGrowing*(*X*)→*Danger*(*Crash*).
CrashAt(*X*)∧*Danger*(*Crash*) →¬*UnBlocked*(*AB*).
Guidance actions:
Danger(*Crash*) ∧ *UnBlocked*(*AC*) → *TurnAround* ∧ *MoveAccelerate*(*CA*).
Safe(*Crash*) ∧*UnBlocked*(*BC*) →*MoveAccelerate*(*CB*).
Danger(*Crash*) ∧ ¬*UnBlocked*(*AC*) →*Stop* ∧ *AskforHelp*.

According to above perturbation handling mechanism, if Jack find a crash at time step *t*=5 and the crash is too wide to cross, then Jack should return to location *A*.

SpectFrontRoute, 1<*t*<5.
CrashAt(*X*), *X*∈*AB*, *t*=5.
CrashWidth(*Width*)∧*Width*>5, *t*=6.
CrashAt(*X*)∧*CrashWidth*(*Width*)∧*Width*>5→*Danger*(*Crash*), *t*=7.
CrashAt(*X*)∧*Danger*(*Crash*) →¬*UnBlocked*(*AB*), *t*=8.
Contra(*UnBlocked*(*AB*), ¬ *UnBlocked*(*AB*), t=9.
Delete(*UnBlocked*(*AB*)), *t*=10.
SpectBackRoute, *t*=11.
Danger(*Crash*) ∧ *UnBlocked*(*CA*) → *TurnAround* ∧ *MoveAccelerate*(*CA*), *t*=12.

In this process, KB is continuously updated, such as ¬*UnBlocked*(*AB*) is added to KB, the expectation changes to *MoveAccelerate*(*CA*), and so on.

6. Conclusion

Base on combining metacognitive loop with improved active logic, this paper proposed an efficient robot commonsense reasoning framework. Among metacognitive loop as the top regulating system play a key role in monitoring, evaluating, and guiding the reasoning process. Improved active logic is able to resolve contradicted knowledge reasonably, to keep every historical reasoning step as the reasoning is time situated, and to continuously update knowledge in KB. However in our example, we did not consider many kinds of perturbations, also the evaluation methods and guidance actions are very limited, therefore, Jack is not able to deal with unknown perturbations efficiently. Fortunately despite lack of these explicit perturbation handling methods and in the simplest case, suppose Jack could not deeply evaluate some perturbations, he can ask us for help or stop from executing missions in a modest way. Our research group is now focusing on human computer interactions (HCI) using a MCL based methodology, and some state-of-the-art reinforcement learning methods such as natural actor-critic (NAC), and simulating a virtual robot to learn itself and the world. Our long term research aims to construct a human-level and flexible robot who can learn, reason, memory, thing, percept, and interact with the world.

Acknowledgements

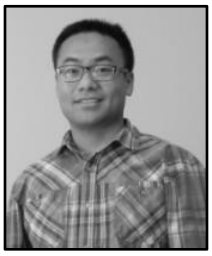
This research was supported by International Exchange Program of Harbin Engineering University for Innovation-oriented Talents Cultivation, National Natural Science Foundation of China (Grant No. 61100007, 61100081) and Fundamental Research Funds for the Central Universities (HEUCFT1102, HEUCF041207).

References

[1] M. Anderson and D. Perlis, "Logic, self-awareness and self-improvement: The metacognitive loop and problem of brittleness," Journal of Logic and Computation, vol. 15, no. 1, **(2005)**, pp. 21-40.
[2] M. Anderson, T. Oates, W. Chong, and D. Perlis. "The metacognitive loop I: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance," Journal of Experimental and Theoretical Artificial Intelligence, vol. 18, no. 3, **(2006)**, pp. 387-411.
[3] Tongchun Du, Michael T. Cox, Don Perlis. "From Robots to Reinforcement Learning". **(2013)** International Conference on Tools with Artificial Intelligence, Washington DC.
[4] Don Perlis, Michael T. Cox, Michael Maynord, et al. A Broad Vision for Intelligent Behavior: Perpetual Real-World Cognitive Agents [J]. Advances in Cognitive Systems 2 **(2013)** 1-18.
[5] Richard S. Sutton, Andrew G. Barto. Reinforcement Learning: An Introduction,http://www.cs.ualbert.ca/%7Esutton/book/the-book.html (1 di 4)22/06/**2005** 9.04.27
[6] Thorben Ole Heins. A Case Study of Active Logic. Report.
[7] Preeti Bhargava, Micheal T. Cox, Tim Oates. The Robot Baby and Massive Metacognition: Future Vision.**(2012)**
[8] Tongchun Du, Don Perlis, Michael T. Cox, Jared Shamwell, Tim Oates. "From Robots to Reinforcement Learning." **(2013)** ICTAI proceeding
[9] Jared Shamwell, Tim Oates, Preeti Bhargava, Michael T. Cox, Uran Oh, Matthew Paisner, Donald Perlis, "The robot baby and massive metacognition: Early steps via growing neural gas." Proceedings of the IEEE Conference on Development and Learning - Epigenetic Robotics **(2012)** (ICDL/EpiRob)
[10] M. Anderson, D. Josyula, Y. Okamoto, and D. Perlis, "Time-situated agency: active logic and intention formation", 25th German Conference on Artificial Intelligence, **(2002)**.
[11] H. Haidarian, W. Dinalankara, S. Fults, S. Wilson, D. Perlis, M. Schmill, T. Oates, D. Josyula and M. Anderson, "The Metacognitive Loop: An Architecture for Building Robust Intelligent Systems", Proceedings of the AAAI Fall Symposium on Commonsense Knowledge (AAAI/CSK'10) **(2010)**, Arlington, VA, USA, November 11-13.

Authors

Kejun Wang, he received the Master's degree in Automatic control theory and applications from Harbin Shipbuilding Engineering Institute in 1987, and PhD degree in Ships and marine equipment and systems from Harbin Engineering University in 1995. His research interests are biometric features recognition, bioinformatics and biometric, and fitness things.



Tongchun Du, he received his Bachelor's degree from Harbin Engineering University in 2009 and is a PhD student majored in pattern recognition and intelligent system up to now. His research mainly focuses on machine learning and artificial intelligence.

Xiaofei Yang, he received his Bachelor's degree from Heilongjiang University in 2006 and is currently a PhD student majored in pattern recognition and intelligent system in Harbin Engineering University. His research involves finger vein recognition and biometric features fusion theory, etc.