Modeling Human Intelligence as A Slow Intelligence System

Tiansi Dong, Member of ACM, tdong@acm.org

Abstract— **[Chang, 2010] argues that slow intelligence is the property shared by surprisingly large number of intelligent systems, and presents examples in scenarios of multimedia data query and mission control in crisis management systems. This paper strengthens this argument by showing that human infant's object tracing can be understood as a slow intelligence and be simulated in this framework. We first introduce an experiment on infant's capability of object tracing, and present two rules in psychology which govern object tracing result, namely, the rule of minimal spatial transformation and the rule of categorical continuity. We review related works and reduce object tracing problems into the problem of object mapping. We show that the following capabilities can be simulated in the slow intelligence framework: the capability to acquiring knowledge about one scene, and the capability to mapping object between scenes. We implemented a simple slow intelligence system in C# to simulate the experiment results presented at the beginning of the paper. Some discussions and outlooks are presented.**

Index Terms— **slow intelligence system, object tracing, simulation**

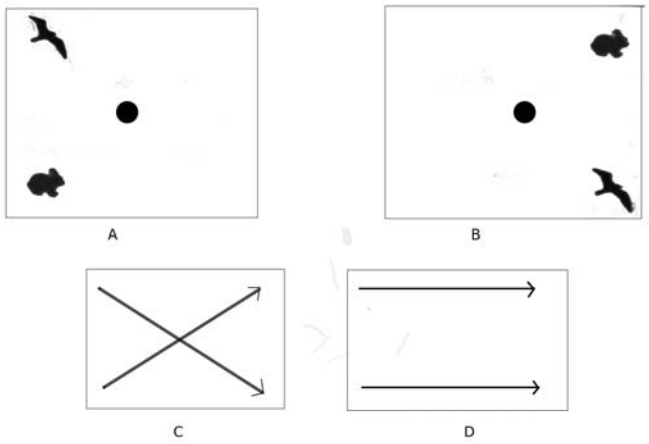

Fig. 1. Two ways to observe the changing from Panel A to Panel B lead to different interpretations. One way follows minimal changes of object categories (Panel C), the other follows minimal changes of locations (Panel D)

I. Introduction

Though adults can play football, pick out fresh apples in grocery, and enjoy films, newly born infants have difficulty in tracking moving objects. [Carey, 2009] presents an interesting example about object tracing as follows: at first you look at Panel A, then close your eyes for 5 minutes; when you open the eyes again, you see Panel B (instead of Panel A). In this case, you will understand that the bird has flied from upper-left to bottom-right and the rabbit has moved from bottom-left to upper-right. Imagine the situation as [Carey, 2009, p.72-73] described as follows: *imagine that the center is new a fixation point, and Panels A and B are projected one after the other onto a screen while you maintain fixation on the common point* and *the timing of the stimuli supports apparent motion*. What would happen? You will see a bird moves from upper-left to upper-right and turns out to be a rabbit and a rabbit moves from bottom-left to bottom-right and turns out to be a bird, as illustrated in Figure 1. Two rules were drawn by psychologies for object tracing: one is that only objects in the same category are traced; the other is that objects are so traced that the spatial difference between the two scenes is minimal. One privileges spatial information leading to the object racing result in Panel C, the other privileges the category (or property) information, leading to the object tracing result in Panel D. There are lots of research works carried out by psychologists to explore infants' object tracking problems, see [Nakayama et al., 1995] and [Carey, 2009] for a review. In this paper, we focus on the computational simulation of object tracing.

The philosophy we follow is a developmental perspective, such that adults' intelligence system is developed from infants', and infants' intelligence system is developed from the state that they even cannot trace moving objects. This also explains why we are interested in the possibility to simulate object tracing – It is one of the early stages towards simulating human intelligence. [Chang, 2010] observes that almost all of the intelligence systems in nature, including a rose, have, in his terminology, *slow intelligence*. He explains its functional definition as follows: it is the capability to solve problems by trying different solutions, to adapt to different situations and to propagate knowledge; with slow intelligence an agent may not perform well in the short run but continuously learns to improve its performance over time. A *slow intelligence system* (SIS) consists of function models as follows: enumeration, propagation, adaptation, elimination, and concentration. The framework of slow intelligence system is illustrated in Figure 2. he outlines two applications of SIS: query processing in multimedia database, and emergency management systems. If his argument is sound, the SIS frame shall be used to simulate human cognitive activities.

The aim of this paper is to show the possibility of simulating an object tracing problem in the framework of slow intelligence system (SIS). The rest of the paper is structured as follows: Section 2 analyzes the object tracing problem and briefly reviews some related works; section 3 and section 4 show that knowledge acquisition from static scene and between scenes can be both understood in the slow intelligence framework; section 5 presents a simple C# implementation of a SIS which simulates the experiment results of Figure 1.

II. Object tracing problem

A universe without objects . . . is a world in which space does not constitute a solid environment but is limited to structuring the subjects very acts; it is a

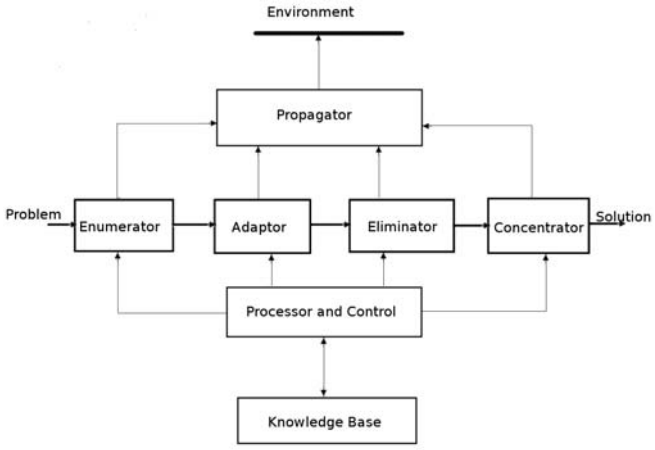

Fig. 2. The framework of the advanced building block of a slow intelligence system; the same framework without knowledge-base is a basic building block of a slow intelligence system

> *world of pictures each one of which can be known and analyzed but which disappear and reappear capriciously.*
>
> *[Piaget, 1954, pp.3-4]*

Human infants can not see constant object when they are newly born. For a spatial change of an object from location A to B, they will believe two objects, one disappears in location A and one appears in location B, [Piaget, 1954]. The object tracing problem can be stated as follows: an agent perceives a sequence of snapshots of the scenes, how can he integrate them into a spatial-temporal whole, such that location changes of objects would be understood as objects' movements? A typical situation is to see a film: We have no difficulty in tracing objects and constructing continuous spatial-temporal changes among the motion pictures. In this paper we focus on the simplest situation in object tracing: Suppose an agent perceives two different pictures, e.g. Panel A and Panel B in Figure 1, how shall he map objects between the two pictures, so that mapped objects shall be recognized as the same instance and spatial differences between them shall be understood as spatial motion? For the example as illustrated in Figure 1, an agent might understand, *the bird has flied from upper-left to bottom-right*, or *the bird has flied from upper-left to upper-right and turned out to be a rabbit*. This suggests two sub-tasks for object tracing: one is object recognition, e.g. an agent recognizes birds and rabbits, the other is object mapping, either a bird is mapped to another bird located somewhere else, or a bird is mapped to a rabbit. Object recognition can be achieved with one scene, while object mapping is a reasoning process between two scenes. Therefore, we approach to the object tracing problem in two phrases: the first phrase is to retrieve knowledge from one scene, the second is to map knowledge between two scenes. Before to show that both problems can be approached in the framework of slow intelligence system, we review some related works.

A. Knowledge of objects

We look at scenes and recognize objects. Our visual system is capable of recognizing objects from stimuli. Psychological research shows that objects are recognized first at a particular level of abstraction. In particular [Rosch et al., 1976] found that categories within taxonomies of objects are structured such that there is generally one level of abstraction at which humans find it easiest to name objects and recognize them the fastest, namely "basic level category". Basic level of abstraction is the level at which categories carry the most information, possess the highest cue validity, and are, thus, the most differentiated from others. [Jolicoeur et al., 1984] found that every object has one particular level at which contact is made first with semantic memory. This level corresponds to the basic level in most cases. That is, the task of recognizing objects is a task of categorization.

B. Knowledge of static spatial relations

Knowledge of spatial relations has been researched with fruitful results. To name a few, [Tolman, 1948] proved the existence of mental spatial representations by serial experiments with rats. [Piaget and Inhelder, 1948] found that human babies acquire the three types of relations in a specific order: topological relations, orientation relations, and distance relations. [Carey, 2009] noted that infants first make a categorical distinction between contact and non-contact. [Dong, 2005] formalized a qualitative spatial knowledge representation for scenes, in which the connection relation is primitive, and other qualitative spatial relations are developed.

C. Relations between knowledge of scenes

Object tracing problem can be addressed as an object-mapping problem between two scenes. Given knowledge representations of two scenes, how shall be objects in one scene mapped to objects in the other? Two rules adopted in this paper are that the rule of minimal spatial transformation (spatiotemporal information) and the rule of categorical continuity (feature information). With the two rules it is quite easy to explain the result shown in Panel C, Figure 1. To explain the result of apparent motion shown in Panel D, Figure 1. Our hypothesis is that object recognition and spatial mapping are two separate cognitive routines, each corresponds to one rule: If they work in parallel, the routine of spatial mapping would be faster and deliver results before that of object recognition. Or, routine of spatial mapping has the primacy over that of categorization, i.e. [Scholl et al., 1999]. In either case, spatial mapping routine works with un-recognized objects, and later these un-recognized objects are labeled with names.

III. Acquiring spatial knowledge of a scene

In this section we show how to acquire spatial knowledge of a scene within the slow intelligent framework.

[Dong, 2007] argued that three kinds of spatial relations can be developed by the connection relation with three axioms

as follows.

1. $\forall x\ [\mathbf{C}(x,x)]$
2. $\forall xy\ [\mathbf{C}(x,y) \rightarrow \mathbf{C}(y,x)]$
3. $\forall x,y[\mathbf{C}(x,y) \rightarrow \forall \mathtt{z} \exists z[z \in \mathtt{z} \wedge \mathbf{C}(x,z) \wedge \mathbf{C}(z,y)]]$

where x, y, and z are regions in one scene, $\mathtt{z}$ is the category of region z. $\mathbf{C}(x,y)$ is read as 'region x and region y is connected', and means 'one region, x or y, and the closure of the other share a common point', i.e. [Kelley, 1955].

The first two axioms are trivial. The third axiom is the characteristic property of the connection relation, which can be interpreted inside of the slow Intelligence framework (illustrated in Figure 2) as follows: the problem is whether x and y are connected. If an agent with slow intelligence finds one category of region, such that for each region in this category, it cannot connect with both of the two regions, he will conclude that x and y are disconnected from each other. That is, universal and existential quantifications in the logical formula serves as the enumerating process. With $\forall \mathtt{z}$ a SIS agent will enumerate all region category $\mathtt{z}$ and with $\exists z[z \in \mathtt{z}]$ he might still enumerate all regions z in region category $\mathtt{z}$. The enumerating process generates values of variables for the predicates. Interpretations of predicates are determined within applications (the adaptation phrase in SIS). For example, in general topology, two regions being connected is interpreted as *one region and the closure of the other share a common point*. Predicates filled with variable values serve as the elimination phrase, for an agent can make decisions on whether the enumerating process shall continue or the problem is solved. After the problem solving process the SIS agent will concentrate on a simple relation between x and y: being connected, or disconnected.

The near extension of a by e, written as a^e, is defined as the region z such that for every region w, z connects with w, if and only if there is a region e_0 in the same category as e such that e_0 connects both with x and w. This can also be interpreted inside of the slow intelligence framework: firstly, enumerating all possible locations of e, then eliminating those that disconnect from a, at last concentrating on the space of a plus those region e which connects with a.

$$a^e \stackrel{\text{def}}{=} \iota z[\forall w[\mathbf{C}(w,z) \equiv \exists e_0[e_0 \in \mathtt{e} \wedge \mathbf{C}(x,e_0) \wedge \mathbf{C}(e_0,w)]]$$

Look at Figure 3, we know that object a is nearer to object b than to object c. This can be formally defined as that *the near extension of a_1 by object e connects with object b while disconnects from object c, where a_1 is the near extension of object a by region e*. That is, $\mathbf{C}((a^e)^e, b) \wedge \neg\mathbf{C}((a^e)^e), c)$. Therefore, distance comparison can also be interpreted within the slow intelligence framework. Imagine that a slow intelligent agent (SIS) perceives Panel A in Figure 1, as illustrated in Figure 4, it can perform near extensions of the two objects, respectively, and this SIS agent would know that the bird is nearer to the upper-left side than to the bottom-right side, the rabbit is nearer to the bottom-left side than to the upper-right side, as illustrated in Figure 5.

As distance relations can be developed based on the connection relation, [de Laguna, 1922], [Dong, 2008], and orientation relations can be understood as the distance comparison, [Dong and Guesgen, 2008], we conclude that both distance relations and orientation relations can be interpreted in the slow intelligence framework.

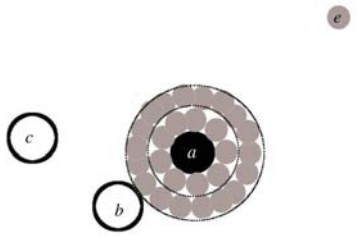

Fig. 3. Knowledge that object C is nearer to Object A than to Object B can be acquired inside of the slow intelligence framework

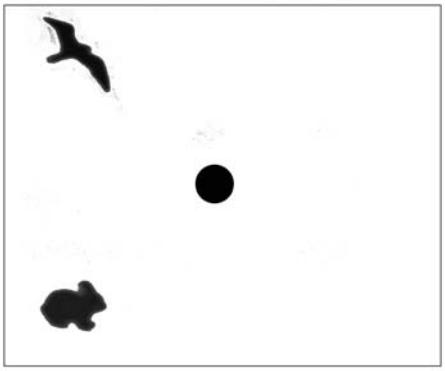

Fig. 4. A slow intelligent system knows neither birds nor rabbits. When it perceives this scene, what it knows is two instances of the unknown object category, one is located upper-left, the other is located bottom-left

IV. Acquiring spatial knowledge between scenes

The spatial knowledge between scenes can be acquired by projecting the second scene onto the first one, and mapping objects in two scenes by minimizing the total cost of spatial transformations within the same object category. For example, we can project the two scenes in Figure 1 together, as illustrated in Figure 6. If we are capable of recognizing birds and rabbits, we will map the bird originally in the first scene to the bird originally in the second scene, and map the rabbit originally in the first scene to the rabbit originally in the second scene. To ease the object reference, we name each object by its category and its original scene name, e.g. <bird,A> refers to any bird from scene A, <rabbit,B> refers to any rabbit from scene B.

Imagine that a slow intelligent agent (SIS) is in a very early stage that it does not know what birds or rabbits look like. When this SIS perceives Panel A in Figure 1, as illustrated in Figure 4, it only know there are two unknown objects with the name: <UNKNOWN,A>. When Panel A suddenly change to Panel B, it perceives two unknown objects located differently. Then with the condition of minimizing the total cost of spatial

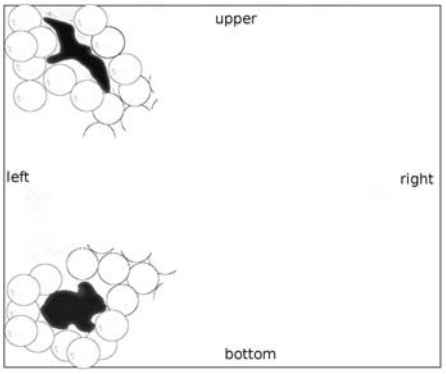

Fig. 5. Location of the two objects can be acquired within the slow intelligence framework, i.e. the near extension procedure

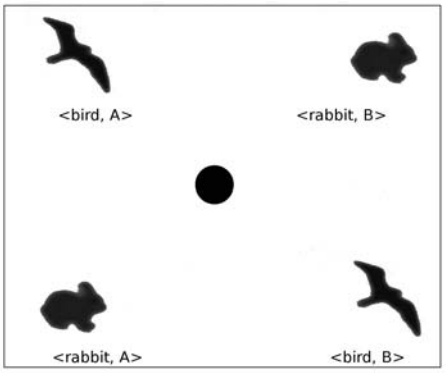

Fig. 6. Project the two scenes in Figure 1 together, objects are named by its category, original scene name

transformation, mapping two objects of the same category in Panel A to the two objects in Panel B would lead to the conclusion that the bird in Panel A is mapped to the rabbit in Panel B and that the rabbit in Panel A mapped to the bird in Panel B, as illustrated in Figure 7.

Let us make the scene-mapping problem clearer and show how it can be approached in the slow intelligence framework. From one scene, an agent acquires knowledge of objects in the scene and spatial relations among them. This knowledge, under certain conditions, can be represented in a graph structure $\mathcal{G} =< V, E >$, the node set $V = \{v_1, v_2, \ldots, v_n\}$ represents objects in the scene, each v_i has two kinds of information of the represented object: its recognized category, v_i.`category`, and its location in this scene, v_i.`location`. The edge set $E = \{e_1, e_2, \ldots, e_m\}$ represents spatial relations between the nodes. For a metrical space E contains $2^{|V|}$ elements[1]. The scene-mapping problem might turn out to be the graph mapping problem: Given two graphs $\mathcal{G}_1 =< V_1, E_1 >$ and $\mathcal{G}_2 =< V_2, E_2 >$, find a mapping $f : V_1 \rightarrow V_2$ such that v and $f(v)$ are of the same category and that the sum of $trans(v_i, f(v_i))$ reaches minimum, $trans(u, w)$ represents the spatial transformation between u and w. The minimum-

[1]In general, the edge set E is not easy to acquire. The reason is that spatial knowledge acquired through perception is neither metrical nor complete, rather qualitative, with distortions, and selective. In this case, E does not contain numerical distance or orientation information between any of two nodes.

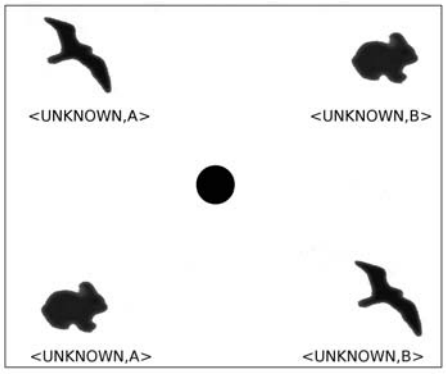

Fig. 7. If a SIS agent can not distinguish birds from rabbits, it will believe that the bird in Panel A moves left and turns out to be a rabbit, and that the rabbit in Panel A moves left and turns out to be a bird

finding task can be approached in slow intelligence framework as the procedures of enumerating, adapting, eliminating, and concentrating. That is, enumerating a possible mapping f_i, adapting to real environment, eliminating f_i if its total transformation is not minimum, and concentrating on the $f_{minimum}$. Based on $f_{minimum}$, a SIS will interpret location difference between mapped objects as movements.

V. An implementation

In this section we present a simple slow intelligence system implemented in C# which simulates two object tracing results in Figure 1. The `RecognizedObject` class represents information of an recognized object, including in which scene an instance is located, to which category this object is classified (recognized), its location in the scene, object index in this scene `Index`, and its object index in the next scene `NexIndex`. `NextIndex = -1` means that this object has no mapped object in the next scene.

```
class RecognizedObject{
  ...

  int sceneId = 0;
  string category = "UNKNOWN";
  int[] location = new int[2];
  int Index = 0;
  int NextIndex = -1;}
```

The `SceneKnowledge` class represents information of knowledge retrieved from a scene. Its members are an Id number `sId`, and a list of recognized objects `sceneObjs`.

```
class SceneKnowledge{
   ....
   ArrayList sceneObjs;
   int sId=-1;}
```

The `ObjectTracing` class contains the main object tracing procedures. Its members are a list of scene knowledge `scenes`, each element of `scenes` is an instance of `sceneObjs`; the maximum length of all `sceneObjs` is stored in `MatrixSize`; `PermutatinMap` is a matrix containing all the permutation of `[0,...,MatrixSize-1]`.

```
class ObjectTracing
{
  ...
  ArrayList scenes;
  ArrayList PermutationMap;
  int MatrixSize;}
```

The enumerating section of the slow intelligence is simulated by `GeneratePermutationMap` procedure. This static procedure generates `PermutationMap` matrix.

```
ArrayList GeneratePermutationMap(int mapSize)
{
    ArrayList lst = new ArrayList();
    ArrayList result = new ArrayList();
    for (int i = 0; i < mapSize; i++)
    {
       lst.Add(i);
    }
    lst = Permutation(lst);
    IEnumerator etr = lst.GetEnumerator();
    while (etr.MoveNext())
    {
     ArrayList ele =
                   (ArrayList) etr.Current;
     result.Add(ele.ToArray(typeof(int)));
    }
    result.Reverse();
    return result;}
```

The elimination part of the slow intelligence system is simulated by `sceneMapping` method. This method takes two `SceneKnowledge` instances as inputs, and try to map the objects between the two scenes by eliminating impossible candidates which are enumerated by the permutation matrix. The main algorithm is as follows:

```
public int[] sceneMapping
       (ArrayList scene0, ArrayList scene1)
{
  int ObjNum0 = scene0.Count;
  int ObjNum1 = scene1.Count;
  RecognizedObject[] ObjArray0 =
       (GetRecognizedObjectsFrom scene0);
  RecognizedObject[] ObjArray1 =
       (GetRecognizedObjectsFrom scene1);

  ArrayList map = new ArrayList();
  map = GetPermutationMap();
  IEnumerator etr = map.GetEnumerator();
  etr.MoveNext();
  int[] tArray = (int[])etr.Current;
  etr.Reset();
  int[] tIndex = new int[0];
  double LeastSpatialTransformation = -1.0;
  while (etr.MoveNext())
  {
    int[] sArray = (int[])etr.Current;
    double currentTransformation = 0;
    int i = 0, s, t;
    RecognizedObject sObj, tObj;
    bool SameCategory = true;
    do{
       s = sArray[i]; t = tArray[i];
       if (s < ObjNum0) sObj = ObjArray0[s];
       else continue;
       if (t < ObjNum1) tObj = ObjArray1[t];
       else continue;
       if (!sObj.SameCatetory(tObj))
       {
          SameCategory = false;
          currentTransformation = -1.0;
          continue;
       }
       currentTransformation +=
        sObj.GetSpatialTransformation(tObj);
    } while ((++i < GetMatrixSize()) &&
             (s < ObjNum0) &&
             (t < ObjNum1) &&
             SameCategory);
    if (LeastSpatialTransformation == -1.0)
    {
       LeastSpatialTransformation
                   = currentTransformation;
       tIndex = sArray;
    }
    else
    {
     if ((currentTransformation
           < LeastSpatialTransformation)&&
         (currentTransformation > 0))
     {
       LeastSpatialTransformation
                    = currentTransformation;
       tIndex = sArray;
      }
     }
    }
    return tIndex;}
```

`ObjArray0` stores the objects in the first scene, `sArray` stores the current permutation of the objects in `ObjArray0`; The permuted objects is mapped to the objects in `ObjArray1`. The permutation will be eliminated if either there is an object pair whose categories are not same, or the total spatial transformation is not minimal. This method returns the permutation of the `ObjArray0` in the form of an integer array. The ith value being j means that the ith object in the first scene is moved to the jth object in the second scene. This information is set to the `IndexInNextScene` member of each object instance.

The whole object tracing procedure is carried out by `doTracing`, listed as follows.

```
public void doTracing()
{
   object[] scns = scenes.ToArray();
   int MaxObjNum = GetMaxinumObjects(scenes);
   SetPermutationMap(MaxObjNum);
```

```
Scene 1:
BIRD with Index 0 located at (1,5)
RABBIT with Index 1 located at (1,1)

Scene 2:
RABBIT with Index 0 located at (5,5)
BIRD with Index 1 located at (5,1)

 In Scene 1:
***BIRD with Index 0 located at (1,5)
 moved to the object with index 1 in the next scene
***RABBIT with Index 1 located at (1,1)
 moved to the object with index 0 in the next scene

 In Scene 2:
This is the last scene.
```

Fig. 8. A slow intelligence system with the capability of recognizing birds and rabbits traces objects moving in Figure 1

```
Scene 1:
UNKNOWN with Index 0 located at (1,5)
UNKNOWN with Index 1 located at (1,1)

Scene 2:
UNKNOWN with Index 0 located at (5,5)
UNKNOWN with Index 1 located at (5,1)

 In Scene 1:
***UNKNOWN with Index 0 located at (1,5)
 moved to the object with index 0 in the next scene
***UNKNOWN with Index 1 located at (1,1)
 moved to the object with index 1 in the next scene

 In Scene 2:
This is the last scene.
```

Fig. 9. A slow intelligence system with NO the capability of recognizing birds and rabbits traces objects moving in Figure 1

```
    for (int i = 0; i < scns.Length - 1; i++)
    {
      int[] IndexInNextScene = new int[0];
      SceneKnowledge sk0 =
                    (SceneKnowledge)scns[i];
      SceneKnowledge sk1 =
                (SceneKnowledge)scns[i + 1];
      IndexInNextScene =
              sceneMapping(
                    sk0.GetSceneObjects(),
                    sk1.GetSceneObjects());
      sk0.SetIndexInNextScene
                          (IndexInNextScene);
    }
}
```

To simulate the first result in Figure 1, we create scene one with the first object `BIRD` located at `(1,5)` and the second object `RABBIT` located at `(1,1)`, and scene two with the first `RABBIT` located at `(5,1)` and the second `BIRD` located at `(5,1)`. The object-tracing result is illustrated in Figure 8. If the slow intelligence system can not distinguish birds from rabbits, that is, we set all the object categories into `UNKNOWN`, the object-tracing result is illustrated in Figure 9.

It is quite easy to see that the computational complexity of this implementation is $\mathcal{O}(n!)$ with regards to the maximum object numbers n in scenes, therefore, it is indeed a *slow* system. One interesting also important argument in [Chang, 2010] is that a slow intelligence system can be developed into a quick intelligence system. We hope the research in this topic would provide a new perspective to the classic NP&P problem.

VI. Conclusions, discussions and outlooks

Object tracing is one of the most important intelligent capabilities which infants developed in the early stage. We show that slow intelligence system (SIS) framework can be applied to simulate the decision process of object tracing, and provide a C# implementation. We argue that SIS framework can be applied to simulate human cognitive system.

In the C# implementation, the knowledge representation of a single scene is simplified. For example, object locations are only represented in quantitative form, i.e., 2-D points. Qualitative or uncertain spatial knowledge shall be included in the future, so that human-style spatial knowledge representations could be simulated.

Future work also includes using SIS framework to simulate more core human cognition, [Carey, 2009], and to connect with existing works in multimedia information processing, e.g. query on spatial-temporal data, [Li and Chang, 2004].

References

[Carey, 2009] Carey, S. (2009). *The Origin of Concepts*. Oxford Univesity Press.

[Chang, 2010] Chang, S. K. (2010). A general framework for slow intelligence systems. *International Journal of Software Engineering and Knowledge Engineering*, 20(1):1–18.

[de Laguna, 1922] de Laguna, T. (1922). Point, line and surface as sets of solids. *The Journal of Philosophy*, 19:449–461.

[Dong, 2005] Dong, T. (2005). SNAPVis and SPANVis: Ontologies for Recognizing Variable Vista Spatial Environments. In Freksa, C., Knauff, M., Krieg-Brückner, B., and Barkowsky, T., editors, *Proceedings of Spatial Cognition 2004*, Lecture Notes in Artifical Intelligence, pages 344–365. Springer, Berlin.

[Dong, 2007] Dong, T. (2007). Towards a Spatial Representation for the Meta Cognitive Process Layer of Cognitive Informatics. In *Proceedings of the 6th IEEE International Conference on Cogntive Informatics*, pages 52–61. IEEE CS Press, Lake Tahoe, California, USA.

[Dong, 2008] Dong, T. (2008). A Comment on RCC: from RCC to RCC^{++}. *Journal of Philosophical Logic*, 37(4):319–352.

[Dong and Guesgen, 2008] Dong, T. and Guesgen, H. W. (2008). A Uniform Framework for Orientation Relation based on Distance Comparison. In *Proceedings of the 7th IEEE International Conference on Cogntive Informatics*, pages 75 – 82. IEEE CS Press, Stanford University, California, USA.

[Jolicoeur et al., 1984] Jolicoeur, P., Gluck, M. A., and Kosslyn, S. M. (1984). From pictures to words: Making the connection. *Cognitive Psychology*, 16:243–275.

[Kelley, 1955] Kelley, J. K. (1955). *General Topology*. Springer, New York.

[Li and Chang, 2004] Li, X. and Chang, S. K. (2004). An Interactive Visual Query Interface for Spatial/Temporal Data. In *Proceedings of International Conference on Distributed Multimedia Systems*, pages 257–262, San Francisco Bay, CA.

[Nakayama et al., 1995] Nakayama, K., He, Z., and Shimojo, S. (1995). Visual surface representation: A critical link between lower-level and higher-level vision. In Kosslyn, S. and Osherson, D., editors, *Visual cognition: An invitation to cognitive science*, pages 1–70. MIT Press, Cambridge, MA.

[Piaget, 1954] Piaget, J. (1954). *The Construction of Reality in the Child*. Routledge & Kegan Paul Ltd.

[Piaget and Inhelder, 1948] Piaget, J. and Inhelder, B. (1948). *La représentation de l'espace chez l'enfant*. Bibliothèque de Philosophie Contemporaine, Paris: PUF. English translation by F. J. Langdon and J. L. Lunzer in 1956.

[Rosch et al., 1976] Rosch, E., Mervis, C. B., Gray, W., Johnson, D., and Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, 8:382–439.

[Scholl et al., 1999] Scholl, B., Pylyshyn, Z., and Franconeri, S. (1999). When are sptiotemporal and featural properties encoded as a result of attentional allocation? *Investigative Ophtalmology and Visual Science*, 40:797.

[Tolman, 1948] Tolman, E. C. (1948). Cognitive Maps in Rats and Men. *The Psychological Review*, 55(4):189–208.