The N–Dimensional Projective Approach as a Tool for Spatial Reasoning

Jorge Pais
Instituto Superior de Engenharia de Lisboa
Instituto de Sistemas e Robótica
Instituto Superior Técnico*
jpais@isel.pt

Carlos Pinto-Ferreira
Instituto de Sistemas e Robótica
Instituto Superior Técnico*
*Av. Rovisco Pais 1, Lisboa, Portugal
cpf@isr.ist.utl.pt

Abstract

In this paper, we describe the n-dimensional projective approach *as a hierarchical and modular architecture with a processing mechanism that underlies both spatial backtracking* and *multiple physical properties of entities. Also, it is shown in Euclidean space how cognitive activities of an agent are improved in terms of* flexibility *(e.g. alternative solutions),* reliability *(e.g. error recovery) and* performance *using this approach.*

1. Introduction

Revisiting the *spatial reasoning* area some research work concerned about one [1] and two dimensional spaces [2] was done. But, with respect to three-dimensional [3] or higher-dimensional spaces few work has been developed. In *qualitative spatial reasoning* area some work has been developed independently of space dimensionality [4] but its *effectiveness* decreases exponentially in order to the number of domain entities. Notwithstanding this drawback, some reasoning techniques were developed to improve the effectiveness of the reasoning process [5].
The *n-dimensional projective representation* is based on simple geometrical concepts [6] that offer an effective reasoning process in a *constrained Euclidean space* [7].
In this paper, section 2 describes the *n-dimensional projective approach*. Section 3 provides how multiple physical properties of entities and spatial backtracking can be designed. Examples are shown in section 4. Finally, section 5 describes the conclusions.

2. The N-Dimensional Projective Approach

This approach is a hierarchical and modular architecture, where lower levels define the *projective representation* and higher levels stablish the *projective reasoning process*.

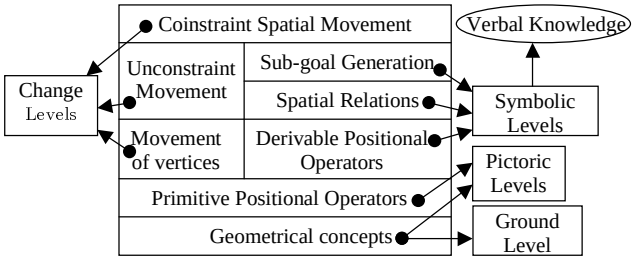

Figure 1: The *n-dimensional projective* architecture.

2.1. Geometrical Concepts –Ground Level

The *ground level* defines the projective representation foundations that are based on both kinds of concepts the topological like $region$ and the geometrical like *projective axis, projective region vertex and projective axis vertex* [6]. Each domain representation includes as many regions as spatial entities that exist in real world $R = \{r_1, r_2, \ldots, r_k\}$. A n-dimensional space S_n is defined by an ordered set of n *projective axis* $S_n = \{A_1, A_2, \ldots, A_N\}$. Each projective axis is an ordered set of *projective axis vertices* $A_i = \{\vartheta_1^i, \vartheta_2^i, \ldots, \vartheta_m^i\}$. Given each projective axis vertex must be a non-empty set of projective region vertices $\vartheta_d^i = \{V_{s,1}^i, V_{e,2}^i, \ldots\}$.

2.2. Primitive Positional Operators

The *primitive positional operators* {<<, ≡} define a minimal model of the world upon a projective axis A_i and they have the following meaning: $V_{s,1}^i << V_{s,2}^i$ iff $V_{s,1}^i$ is closer than $V_{s,2}^i$ from the projective axis origin; $V_{e,1}^i \equiv V_{e,2}^i$ iff both end vertices of regions r_1 and r_2 are equidistant from the projective axis origin.

2.3. Derivable Positional Operators

The *derivable positional operators* define the first symbolic level of our architecture. These operators must be responsive to introducing spatial semantic to the projective representation and they are asserted using the primitive positional operators as described in [7].

2.4. Spatial Relations

This level includes the spatial relations set {OutsideLeft, OutsideRight, OutsideLeftCoincident, OutsideRightCoincident, CompletelyCoincident, CompletelyInside, InsideLeftCoincident, InsideRightCoincident, OverlappedLeft, OverlappedRight} applied to a region which are presented formally in [6]. One of the most important purposes of this level is to provide a decoding from verbal knowledge into a fairly projective geometrical concepts and the opposite.

2.5. Sub-Goal Generation

This level generates sub-goals reachable from the current state respecting *spatial constraints*. The spatial constraints are understood as physical properties of entities in the environment. For example, an embodied system either could acquire (e.g. high reflections on signal sonar could signify an *impenetrable* entity, high temperature detection could imply untouchable entities) or could integrate (e.g. white color could mean an *impenetrable* wall, red color could be an untouchable fire) this sort of knowledge. This change level provides a real-time generation of consistent sub-goals based on a method named *projective sub-goal generation* (PSG). The PSG method classifies each projective axis using three parameters, *current state*, *transition* from current state to goal state and *goal state*. Each one of these parameters could take two different values for each physical property, violation(V) or non-violation(N). The evaluation result for each projective axis determines the set of sub-goals reachable from the current state respecting spatial constraints as is shown in Table 1.

Current State	Current → Goal	Goal State	Sub-goal State for each Projective Axis
N	N	N	Goal State
N	N	V	Impossible
N	V	N	Goal or Current State
N	V	V	Goal or Current or a Non-Violation Goal
V	N	N	Goal State
V	N	V	Goal or Permutations of the Current State
V	V	N	Goal State
V	V	V	Goal or Permutations of the Current State

Table 1: The projective sub-goal generation (PSG) method.

A resumed description of the PSG method follows. The conditions (N, N, N), (V, N, N) and (V, V, N) describe topological configurations where the goal state must be the next sub-goal. The condition (N, N, V) is an inconsistent condition that never happens in a consistent domain because it is impossible to go from a non-violating state to a violating state without a property violation. With respect to the (N, V, N) condition, the method evaluates the topological configuration over all other projective axis. If there exists at least one projective axis that does not happen a property violation then the goal state is the next sub-goal. Otherwise is the current state. The condition (N, V, V) refers a ternary solution situation depending on the projective axis topological configuration. If all other projective axis violate the property then the solution is the current configuration. If there exists at least another non-violating axis the sub-goal is a *non-violating goal state*. It means a topological configuration that respects the distribution of regions in the goal state but without violating the property. The third solution is when all axis share either a non-violating goal state or a goal state then the next sub-goal is the goal state. The critical conditions (V, N, V) and (V, V, V) do not provide any clue to solve the property violation. Thus, it can be solved using a complete sub-goal generation based on permutations among the regions responsible for the violation. And the combination between each one of these permutations with the non-violating regions. The generation of sub-goals based on permutations might be interpreted as a breadth step into the changing process that essentially should be based on depth steps to find out solution paths with $effectiveness$. However, breadth steps can be transformed in depth steps if the system generates one permutation at a time and memorizes these points on the changing process as backtracking points to return to them later.

A property violation detection depends on geometrical characteristics of the physical property. Two spatial properties of entities are considered in our system, *untouchable* and $impenetrable$. All algorithms for detecting the violation or the non-violation of these two physical properties can be found in [7].

2.6. Movement of Vertices

Just two atomic movement operators generate change over each projective region vertex along each projective axis: *MoveVertexLeft* and *MoveVertexRight*. The algorithms of these two operators were described in [7].

2.7. Unconstrained Movement

The one-dimensional unconstrained movement algorithm takes advantage using levels, the *spatial relations* and *the movement vertices*, defined on precedent sections. A detailed description of this algorithm such as a study about its complexity can be found in [7]. The *n-dimensional algorithm NDimProjectiveMove* just requires executing the one-dimensional algorithm so many times as the number of projective axis existing on domain model.

2.8. Constraint Spatial Movement

This level uses both sub-goal generation level and unconstraint spatial movement level. The key idea (see figure 2) is to give to the sub-goal generation level a postponing pair of states to get a complete sub-goal plan. Starting with the initial and final states to get the first sub-goal. After that, the process repeats with the first sub-goal and the final state to get a second sub-goal and so on until get the final state as sub-goal. This postponing generation of sub-goals is done by the $GetPlan$ function that returns either a complete sub-goal plan between two states or an empty plan in case of fail. After that, this level provides consecutive pairs of sub-goals to the $unconstraint\ movement\ level$, which carry out simple vertex motions using the *NDimSpatialMove* function..

```
ConstraintSpatialMove(InitialState, FinalState)
{ Plan= GetPlan(InitialState, FinalState);
  WHILE (Plan≠∅) NDimSpatialMove(RemoveFirstSubGoal(Plan),
                                 GetFirstSubGoal(Plan)); }
```

Figure 2: The algorithm of constraint spatial move.

3. New Spatial Mechanisms

3.1. Multiple Physical Properties

A physical property is respected using the PSG method but when an entity shares distinct physical properties, a relevant question emerges: how can an autonomous artificial system solve the interaction among entities? To solve this problem, we found out a practical and empirical solution that gives

topological shorter paths. Consider N physical properties, represented as $\{P_1, P_2, \ldots, P_n\}$, the empirical solution goes through applying the PSG method in cascade N times to each one of those P_i. To find out paths topologically shorter is convenient to do a sort of all properties in terms of spatial constraints such that P_{i-1} is lesser constraint than P_i, represented as $P_{i-1} \prec P_i$. Then, considering this spatial constraint relations among all properties such that $P_1 \prec P_2 \prec \ldots \prec P_{n-1} \prec P_n$, the system finds out the topological closer sub-goal from the current description cascading the PSG method as shown in Figure 3.

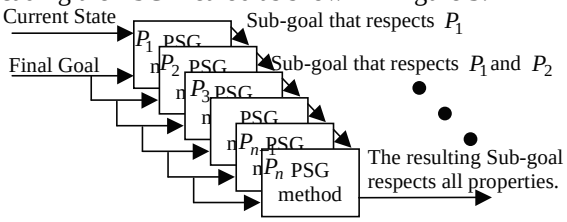

Figure 3: The method to obtain a non-violating sub-goal.

3.2. Spatial Backtracking

Two questions arrive about this important issue. What is spatial backtracking? How can an artificial system implement it? Spatial backtracking could be defined as a spatial reasoning technique that allows an artificial system to obtain alternative solutions to the same topological problem. Particularly, a complete spatial backtracking system for *one-dimensional* space generates from *k* violating entities a maximum of *k factorial* different solutions, that is to say, all spatial possibilities to distribute *k* entities within the space. Generally, in a *n-dimensional* space the dimensionality of the problem increases exponentially in order to *n*, thus the resulting complexity is $(k!)^n$ that defines a very hard NP-complete problem.

The second question is much more complicated to solve because it is a practical problem and in a n-dimensional space any artificial system must not generate all solutions, because all together are computationally impossible of memorizing and manipulating at the same time. However, it has a practical solution that consists in each computational time manipulating and memorizing just one of all solutions and postponing the process always that the user asks for one different solution. This postponing process requires a special architecture that supports *a spatial deduction* mechanism. In Artificial Intelligence (A.I.) any architecture able to give alternative solutions to the same problem should be based either on a search-based mechanism or on a deductive-based mechanism. The projective deduction mechanism is essentially deductive and it is implemented using two stacks (see figure 4). One stack is named the *sub-goal stack* and it memorizes the sequence of sub-goals generated by the successive application of the PSG method to go from an *initial state* to a *final state*. This stack provides the system avoiding loops in terms of the sub-goals plan. In each time that is generated a sub-goal, which already exists in this stack, the system empties the stack until it finds out the sub-goal that generates the first occurrence of the repeated sub-goal.

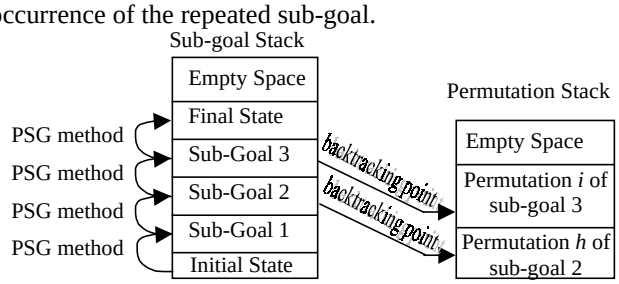

Figure 4: Two stacks underly the spatial backtracking.

Generally, the first sub-goal remained in the sub-goal stack has an entry in another stack that we call *permutation stack*. The permutation stack is responsible for memorizing the last permutation of the violating regions that are associated to the sub-goal. This last permutation defines the topological order that the violating-regions are placed in the last generated sub-goal descendent. With this design, considering *k* violating regions the system is able to generate *k!* topological descriptions that correspond each one of them to one different sub-goal descendent. This postponing process terminates for a sub-goal when its all permutations had been generated. This end process is easily implemented by checking the ordered of *k* violating regions. If this set is ordered the permutation process is complete and the system removes the sub-goal from the sub-goal stack.

Another situation of backtracking is when the system by its constraint spatial movement level gives one sub-goal solution path and another solution will be required to this level. In this case the sub-goal stack is empty until the system finds out the first one sub-goal that are simultaneously inside of two stacks because it identifies a backtracking point and after that the reasoning process starts again.

4. Examples

Below are two examples from classical A.I.. In both cases, the projective system is not provided with knowledge about the domain. It does not know neither what kind of *operators* are allowed in each domain nor it cares about interpretations in the domain. Thus, the only information given to the system is the initial and final topological descriptions and the physical properties of entities.

4.1. The Puzzle Problem

The *puzzle* problem here presented is characterized as follows. It has four tiles and each tile is impenetrable by the others and the initial and final spatial topologies are illustrated in Figure 5.

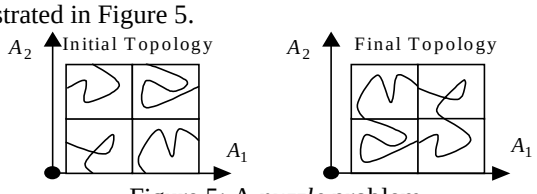

Figure 5: A *puzzle* problem.

The first sub-goal solution plan given by the architecture to be executed is shown in figure 6. But the projective architecture gives tens of different to this kind of problems.

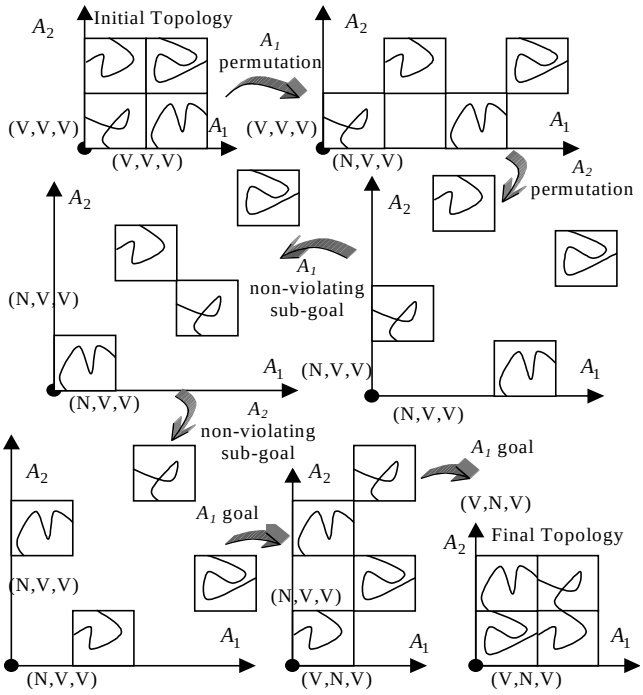

Figure 6: The first sub-goal plan solution.

Figure 6 depicts a sub-goal solution path enough elaborated and complex considering that it does not have any knowledge about the domain. However it is real and proves the effective capability of the projective reasoning system in solving real-world spatial problems just using its deductive spatial system.

4.2. The Monkey and Bananas Problem

Revisiting this classical A.I. problem but from a viewpoint of an embodied system that could be an artificial monkey that likes bananas. Thus, the problem is stated as the monkey being in a closed room with a chair on the floor and with a bunch of bananas hanging on to the ceiling that he wants to eat. If the monkey had a mapping system able to translate from its perception to a spatial *projective representation* then he could use the *n-dimensional projective approach* to solve its problem.

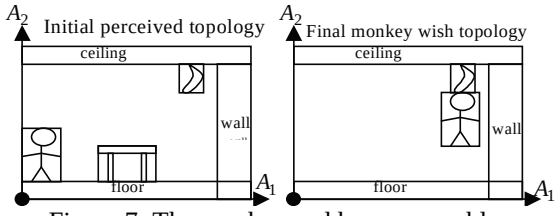

Figure 7: The monkey and bananas problem.

The left side of figure 7 illustrates the initial perceived topology by the monkey and the right side illustrates its wish topology (getting the bunch of bananas).

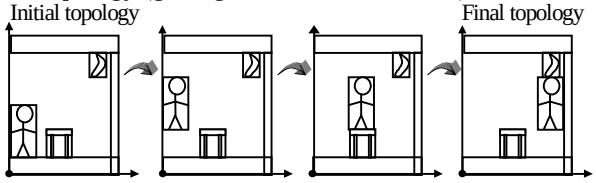

Figure 8: The first sub-goal solution.

The projective approach gives two possible solutions. In Figure 8, the monkey jump on the chair and after that trying to pick the bananas jumping from the chair to the banana's place. For any practical problem, if this plan is not possible to realize by the monkey another alternative solution is shown in figure 9. It provides as monkey actions the following sequential steps: i. It jumps on the chair; ii. It remains on the chair until the plan is over; iii. It should move both the chair and itself from the current place until the place under bananas for the monkey picks them.

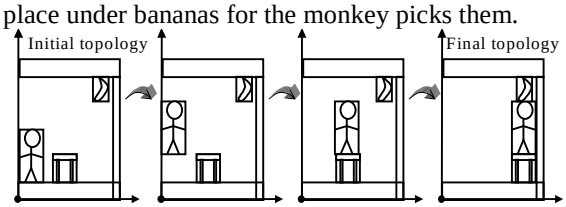

Figure 9: The second and last solution.

These only two sub-goal solutions shown in figures 8 and 9 are not all possible solutions. From one point of view, it demonstrates the incompleteness of the projective architecture. But from another point of view, we prove the usefulness of this architecture to getting real-time solutions in solving real-world problems.

5. Conclusions

This paper describes briefly the *n-dimensional projective* approach as a hierarchical and modular architecture. Also, it introduces two new spatial concepts in the approach – multiple physical properties and spatial backtracking. From these two spatial concepts emerge a very hard NP-complete problem but a new deductive mechanism solves these emerging difficulties preserving two important issues, $effectiveness$ and $computational\ adequacy$. Examples in $puzzle$ and in the *monkey and bananas* problems illustrate promising results in solving real-world problems.

References

[1] Allen, J.F. 1991. Time and Time Again: The Many Ways to Represent Time. In *International Journal of Intelligent Systems* 6, 341-355.

[2] Hernandez, D. 1991. Relative Representation of Spatial Knowledge: The 2D Case. *Cognitive and Linguistic Aspects of Geographic Space*, 373-385, Kluwer Publishers, Netherlands.

[3] Coenen, F.; Beattie, B.; Shave, M.; Bench-Capon, T.; Diaz, B. 1988. Spatial Reasoning Using the Quad Tesseral Representation. *Artificial Intelligence Review* 12, 321-243, Netherlands.

[4] Cui, Z.; Cohn, A.; Randell, D. 1992. Qualitative Simulation Based on a Logical Formalism of Space and Time. In *Proceedings of AAAI92*, AAAI Press.

[5] Pais, J.; Pinto-Ferreira, C. 1998. Search Strategies for Reasoning about Spatial Ontologies. In *Proceedings of 10th IEEE International Conference On Tools with Artificial Intelligence* ICTAI98, pp. 418-422, Taiwan.

[6] Pais, J.; Pinto-Ferreira, C. 2000. Qualitative Spatial Reasoning using a N-Dimensional+ Projective Representation. *In Proceedings of 15th European Meeting on Cybernetics and Systems Research* EMCSR2000, pp. 769-774, Austria.

[7] Pais, J.; Pinto-Ferreira, C. 2000. Spatial Representation and Reasoning using the N-Dimensional Projective Approach. In Technical Report WS-00-08 of the AAAI2000 *Workshop on Spatial and Temporal Granularity*, pp. 79-82, Austin, USA.