Robot navigation combining the Gradient Method and VFF inside JDE architecture [*]

José María Cañas
Grupo de Robótica
Univ. Rey Juan Carlos
ESCET
28933 Móstoles
jmplaza@gsyc.escet.urjc.es

J.Raúl Isado
Grupo de Robótica
Univ. Rey Juan Carlos
ESCET
28933 Móstoles
jraul@gsyc.escet.urjc.es

Lía García
Ing. Sistemas y Automática
Univ. Carlos III
Esc. Politécnica Superior
28911 Leganés
lgperez@ing.uc3m.es

Abstract

The integration of several behaviors into a single versatile robot is still an active research topic, mainly focused on architectural issues. This paper presents our architecture JDE and how it has been applied to solve the robot navigation combining two well known techniques: Gradient Path Planning and Virtual Field Forces. Deliberative and reactive components are merged together in the behavior system, in a different way from layered hybrid approaches. Some experiments with the composed behavior are also described.

1 Introduction

Behavior generation is a very complex issue, even harder in multi-goal robots. In order to get useful applications in the near future, like service robots or personal assistants, we need to improve their behavior systems. More than on the specific control or perception algorithms, a significant part of such improvement lies on the robot architecture. The architecture integrates all the robot capabilities and determines most of the robustness and flexibility of the system. It has been an active area for a long time, but more research is still needed to reach an acceptable performance.

Several architectures have been historically proposed for behavior generation in robots. Dynamic and uncertain environments forced the evolution from symbolic AI to reactive and behavior based systems (BBS). BBS were a revolution [3], but have shown poor scalability for complex systems. Hybrid architectures have been predominant since mid 90s [4; 8], mainly those three-tiered ones that add two layers to BBS, usually a sequencer and a deliberator.

Several hierarchies have been explored after the hybrid architectures became the de facto standard. In particular, many reviews of the hierarchy principle have been proposed in last years [2], trying to overcome subsumption limitations. In this way, a novel hierarchical approach named JDE is presented in this paper.

Navigation is an ubiquitous problem for mobile robotics, as long as the robot has to move through its environment in order to perform its goals, whatever they were. For instance, commercial robots like Roomba[1] vacuums a room by following an spiral navigation. Urbano[2] or Minerva robots guide people through the different rooms of a museum without colliding with people neither walls. The autonomous harvesters of Demeter project[3] travel through huge cereal crops following a GPS route.

The goal of this work is to achieve a com-

[*]This work has been funded by Spanish Ministerio de Ciencia y Tecnología, under the project DPI2004-07993-C03-01

[1]http://www.irobot.com/
[2]http://www.disam.upm.es/robotica/Web_robotica/proyectos/Urbano/
[3]http://www.ri.cmu.edu/projects/project_149.html

plete navigation using two well known techniques, Gradient Path Planning and VFF, and combine them inside the aforementioned JDE architecture.

Second section introduces the JDE control architecture foundations. Third section describes the design of the navigation behavior implemented inside JDE architecture. Several tests and experiments have been conducted and they are commented in fourth section. Finally, fifth section presents some conclusions of our work.

2 JDE control and perception architecture

In JDE stance, behavior is considered the close combination of perception and actuacion. Both are splited in small units called schemas [1]. Nothing new so far. JDE proposes the schemas can be combined in a dynamic hierarchy to unfold the behavior repertoire of the robot accordingly to current goals and environment situation [7].

2.1 Schemas

JDE follows the Arkin's definition [1]: "a *schema* is the basic unit of behavior from which complex actions can be constructed; it consists of the knowledge of how to act or perceive as well as the computational process by which it is enacted". In JDE, each schema has a time scale and a state associated. They can be switched on and off at will and they accept some modulation parameters.

There are two types of schemas, perceptive and actuation ones. Each *perceptive schema* builds some information piece about the environment or the robot itself, which we'll call a *stimulus*, and keeps it updated and grounded. That's its output. It can take as input the value of sensor readings, or even stimuli elaborated by other schemas. Perceptive schemas can be in SLEPT or in ACTIVE state.

Each *actuation schema* takes control decisions in order to achieve or maintain its own goals, taking into account the information gathered by the associated perceptive schemas. The outputs of an actuation schema are tipically commands to actuators, and can also be the activation of other schemas, both perceptive and actuation ones. Such new schemas are regarded as its children, and the parent schema often modulates them. Actuation schemas can be in several states: SLEPT, CHECKING, READY and ACTIVE, closely related to how action selection is solved in JDE. Control schemas have preconditions.

The algorithm running inside the schema contains all the task-knowledge about how to perceive relevant items or how to act, whatever technique be used. JDE architecture only imposes the interface: selective activation (on-off) and parameters for modulation, as long as it affects the way all the pieces are assembled together into the system. Concurrent continuous (iterative) execution is assumed, where each schema actively maps its inputs into its outputs.

2.2 Hierarchy

All awake schemas (CHECKING, READY and ACTIVE) run concurrently, similar to the distribution found in behaviour-based systems. To avoid incoherent behaviour and contradictory commands to actuators JDE proposes hierarchical activation as the skeleton of the collection of schemas. It also claims that such hierarchical organization, in the ethological sense, provides many other advantages for roboticists like bounded complexity for action selection, action-perception coupling and distributed monitoring. All of them without losing the reactivity needed to face dynamic and uncertain environments.

In JDE there is hierarchy as long as one schema can activate other schemas. An actuation schema may command to actuators directly or may awake a set of new child schemas. These children will execute concurrently and they will in conjunction achieve the father's goal while pursuing their own. Actually, that's why the father awoke such schemas, and not others. A continuous competition between all the actuation siblings determines whether each child schema will finally get the ACTIVE state or remains silent in CHECKING or READY state. Only the winner, if any, passes to ACTIVE state and is allowed to send commands to the actuators or spring their own child schemas. The father activates the perceptive schemas that provide the information needed to solve the control competition between its actuation children and the information needed for them to work and take control decisions. This recursive activation of perceptive and actuation schemas conforms a schema hierarchy.

Once the father has awaken their children it keeps itself executing, continuously checking its own preconditions, monitoring the effects of its current kids, modulating them appropriately and keeping them awake, or maybe changing to other children if they can face better the new situation. So the hierarchy is dynamic.

2.3 Action selection

JDE decomposes the whole Action Selection Mechanism (ASM) into several simpler action selection contests. At each level of the hierarchy there is a winner-takes-all competition among all actuation schemas of such level.

Before a given schema wins the control, it has to go through four states. First, when the parent awakes it, it passes from SLEPT to CHECKING state. Second, the schema promotes from CHECKING to READY when its preconditions match current situation. Preconditions define the set of situations in which such schema is applicable and may achieve its goal, which in JDE is named the *activation region* of such schema. And third, typically the preconditions of only one schema will hold and it will move from READY to ACTIVE state. In case of several (or none) READY siblings, the parent is called for choosing a single winner.

For instance, in figure 1 the activation regions of schemas 5, 6 and 7 are displayed. They are defined over the space of possible values for stimuli built by perceptive schemas 3 and 4. Impossible combinations are displayed in shadow. The current situation is a point in such space and it may fall inside an activation region (such schema will promote to READY) or outside. This picture is similar to state-space diagrams in [2].

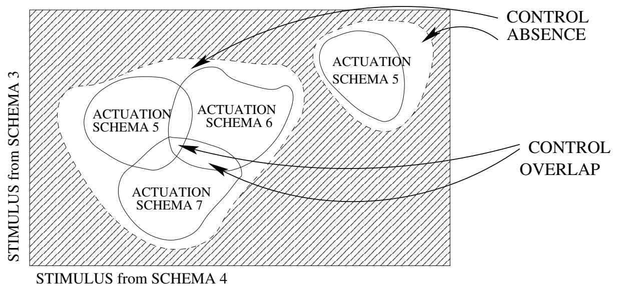

Figure 1: Activation regions of three schemas

The JDE ASM is goal oriented and situated. Goal oriented because the winner lies in the set of schemas already awaken by the father, and no others. And situated because the environment chooses among them the most suitable to cope with current situation. It is also fast: a new action selection takes place at every child iteration, which allows timely reaction to environment changes. It is also flexible, as one schema may have priority over another in some contexts, but not in others.

3 Position based navigation with JDE schemas

Once foundations of JDE have been presented we will comment in this section how we have used it to solve the navigation of our mobile robot combining two diferent techniques inside JDE: Gradient Path Planning (GPP) [5] and Virtual Force Field (VFF) [6]. The GPP will take care of the global navigation, and the VFF will avoid collision with obstacles, even with those not considered in the map. Both have been implemented as JDE-schemas. We have designed and implemented a small hierarchy of JDE-schemas, shown in figure 2, which generates the safe navigation behavior.

The GPP has been designed inside JDE as two separate schemas: Build-Gradient and Follow-Gradient. The first one builds the distance field of GPP taking into account the map of the environment. The second one is an actuation schema which translates the recommendation of the distance field at the current

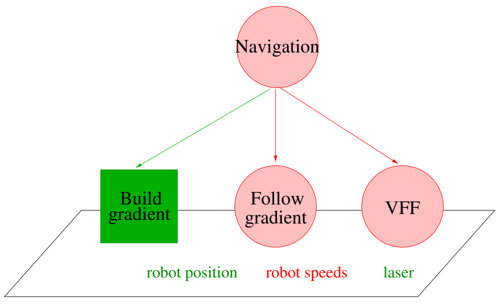

Figure 2: Behavior's schema hierarchy

robot position into motor commands. It reactively executes the implicit plan contained in the distance field, which avoids the static obstacles of the environment and leads the robot towards its destination.

The VFF has been designed inside JDE as a single actuation schema which performs a reactive control based on data from laser sensor and on the recommended orientation from the distance field. It takes into account the obstacles detected by the laser sensor, regardless they appear in the map or not.

3.1 Build-Gradient Schema

This schema is perceptive, it builds the distance field typical from GPP techique from the occupancy map of the environment. The occupancy map is the schema input and the distance field is its output. The distance field requires processing and it is seen in JDE as a stimulus to guide further movement decisions. An stimulus coming from deliberation, not from the sensor data, but a stimulus after all. In addition, such stimulus does not need any further update once it is built, because it depends only on the static map of the environment.

The *distance field* is the sum of two components: an obstacle field and a target field. The first starts at the cells on the obstacles, setting high value to such cells and decreases as the cells are further from them. The second one starts at the destination point with zero value and then expands itself like a wave propagating through the empty space of the environment. The wave front does not pass through the occupied cells. It is a increasing field, points near the robot destination will have lower values than those far from it, the further the higher. Actually, the value of such field at a location is a measurement of the real distance from such point to the destination taking into account the geometry of walls and other obstacles in the environment. A detailed description of the algorithm can be found at [5].

This perceptive schema is activated by the Navigation schema, and then: it creates the obstacle field expanding it up to a certain distance from the obstacles; it creates the target field expanding it until it reaches the current robot position; and it generates the reference route following the exact gradient of the field. The reference route is not used for navigation, just for visualization and debugging purposes.

The position of the destination point is a modulation parameter for Build-Gradient schema. Once it has finished building the field, it activates a flag and waits doing nothing. Its father will read the flag and will sleep it in turn, because its stimulus does not require any further update.

3.2 Follow-Field Schema

This is an actuation schema in charge of piloting the robot to the destination following the gradient of the distance field. The field is considered the input for the schema and its outputs are the translation speed (V) and rotation speed (W) commanded to robot motors. The Follow-Field schema takes movement decisions every 100 ms, complying the iterative nature of schemas in JDE.

The movement decisions are reached in two steps: first, the orientation recommended by the distance field is computed searching for the gradient; second, a fuzzy controller is used to choose the right V and W commands. The gradient is computed searching for the lowest value of the distance field in the 5x5 vicinity of the current position of the robot, as can be seen in figure 3.

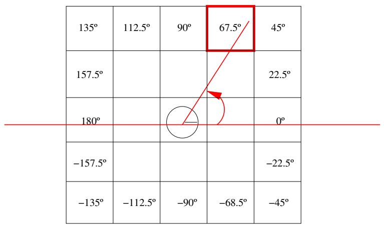

Figure 3: Gradient definition

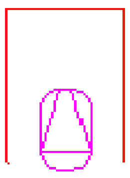

Figure 4: VFF's security area

This schema uses a fuzzy controller to smoothly allign the robot with the field gradient. The controller takes into account the current speeds and the difference between current robot orientation and that of the gradient. It conducts a Proportional control on the rotation speed, and it stops the robot when the angle difference is high. It also considers the value of the field at the current position of the robot, as it indicates how far it is from the destination. When the robot approaches to the target it slows down its translation speed.

With only this schema the robot would always follow the gradient of the distance field, searching for minimum values of the field. The V and W commanded by this schema let the robot to reach destination without colliding with known obstacles, and lead it along a minimum distance path. The distance field implicitly guides the robot away from obstacles (the obstacle component of the field gives high values to dangerous areas) and towards the destination (the target component decreases as the positions are closer to the goal).

The preconditions of Follow-Gradient are set always true.

3.3 VFF Schema

This actuation schema is responsible of avoiding near obstacles. The laser readings and the distance field are the inputs of the VFF schema. The outputs are the V and W commands to robot motors. The VFF schema takes movement decisions every 100 ms, complying the iterative nature of schemas in JDE.

The movement decisions are based on virtual forces. The obstacles detected with the laser sensor cause a repulsive force, and the gradient of distance field generates an attractive force. The robot will move following the direction of the vectorial sum of both forces, reaching a compromise solution between the trend of collision avoidance and that of advancing towards the goal. More details can be found at [6]. Once the global sum has been computed a fuzzy controller is used to translate that into translation and rotation speeds, in a similar way to Follow-Field schema.

Unexpected obstacles are not taken into account by Follow-Field schema, as long as they do not appear in the static map. In VFF schema, both known and unexpected obstacles are considered for movement decisions as long as both are detected with the laser sensor.

VFF schema is activated when any obstacle enters into its security area. This security zone has been defined as a rectangle around the robot, as shown in figure 4. It is longer at the front because collisions in the advance direction are more likely and difficult to avoid than those due to side obstacles.

3.4 Combination inside JDE

The concurrent execution of Build-Gradient, Follow-Gradient and VFF schemas provides the safe navigation of the robot. They are grouped together as children of the Navigation schema, which encapsulates the navigation functionality in a schema interface. The input of the Navigation schema is the map of the environment, its outputs are the preactivation of its children schema and it accepts the destination point as modulation parameter. Its preconditions hold whenever the current robot position is further than a certain proximity threshold from the destination.

If someone wants the robot to reach certain destination, it has to awake the Navigation schema and modulate it through its destination parameter. In our experiments such activation and parameter come from the human operator through the graphical interface of the application.

The Navigation schema perfoms an iteration every 200 ms. When ACTIVE, it initially captures the map of the enviroment in which the robot will operate and stores internally as an occupancy grid. Then it activates Build-Gradient schema and waits until it finishes the distance field. After that, it deactivates the Build-Gradient schema and awakes both Follow-Gradient and VFF actuation schemas, putting them in CHECKING state.

It continually checks its own preconditions and performs the arbitration among its actuation children. The arbitration code lies inside the Navigation schema, but it is called whenever one of its children, Follow-Gradient or VFF, detects the control collision. As we have seen, the Follow-Gradient preconditions are always true, so it will really reach the ACTIVE state whenever VFF keeps itself in CHECKING state. When the security zone is invaded, VFF will pass to READY state and will detect a control collision with Follow-Gradient. Arbitration in Navigation schema has been coded to give priority to VFF over Follow-Gradient, so the first one will upgrade to ACTIVE and the second one downgrade to READY while there is an obstacle in the security area.

This is an example of hierarchy reconfiguration depending on situation. The behavior generation system of the robot usually has Navigation and Follow-Gradient schemas active, but when a close obstacle appears in the robot neighborhood, it reconfigures itself to Navigation and VFF, changing the way the robot behaves.

In addition, Build-Gradient and Follow-Gradient schemas are an example of how deliberation can be inserted inside JDE. Despite its processing may take some time, its use must be reactive, and must provide motor commands like reactive schemas, not abstract data.

4 Experiments

The presented hierarchy of schemas for navigation has been programmed and tested under different conditions. We have ordered the robot towards close, medium and remote destinations inside an indoor scenario, making it to traverse corridors, rooms, open doors, etc., with static and unexpected dynamic obstacles around. The experiments have been done using the SRIsim and the Stage simulators. Both are fully supported in current JDE software implementation. Some of the experiments have been recorded in videos, and they are available at the web[4].

4.1 Typical execution

In figure 5, we can see a typical execution of the application. The grey surface is the distance field, the darker the higher value of the field. The destination is located in the top right room of the figure. The yellow line is the reference route, generated by following the gradient of the distance field. The green pixels are the points where the robot was directed by the Follow Gradient schema, and the orange points are the points where the robot was directed by the VFF schema.

As we can see in figure 5, the robot finally reach its destination and most of the time is guided by Follow-Gradient. Only in narrow passages the VFF schema gains control. The real position of the robot was provided all the time by the simulator, as it is required to compute the right gradient. The average speed along the typical run was 1'3 m/sec, reaching 1'8m/s peaks in obstacle straight free areas like corridors.

4.2 Avoidance of unexpected obstacles

We have also tested the schema hierarchy when unexpected obstacles like people, chairs, bins, etc. disturb the robot navigation. A

[4]http://gsyc.escet.urjc.es/jmplaza/research-navigation.html

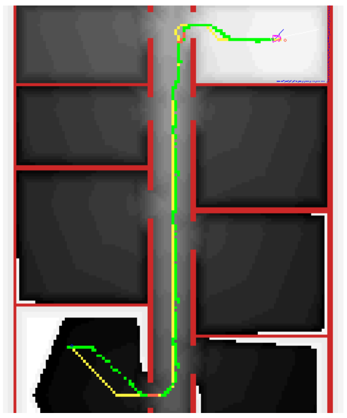

Figure 5: Typical Execution

second robot in the Stage simulator was manually teleoperated to interfere the autonomous robot running JDE.

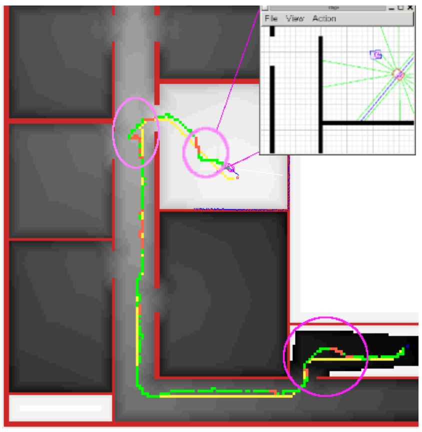

Figure 6: Avoidance of unexpected obstacles

In figure 6 we can see how the autonomous robot deviated from reference route to avoid collision with the unexpected obstacle. Correct alternance between schemas is also visible in the orange trajectory pixels. The VFF schema enters into scene the three times (marked in the figure) when the teleoperated robot interfered the autonomous one. For instance, before leaving the room of the starting point, teleoperated obstacle was placed in front of the autonomous robot, making VFF schema to take control. Once the robot passed over the obstacle, security zone was no longer invaded, and Follow Gradient gets the control back (green pixels), returning to the reference route.

4.3 Navigation refinements

The experiments performed moved us to improve the original algorithm, not its architectural design but the algorithms inside the schemas.

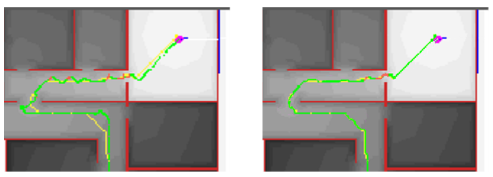

Figure 7: Movement using 3x3 (left) and 5x5 (right) vicinities

First, when computing the gradient of the distance field we tried with the 3x3 and the 5x5 vicinity around the current position of the robot. The 5x5 vicinity provides smoother movements than 3x3, as it lets the robot to react sooner to changes in the field and then to start braking or turning before. This can be seen in figure 7, where 5x5 vicinity (right) generates smoother trajectories than 3x3 vicinity (left). The robot adjusts better to the reference route as it has more time to react.

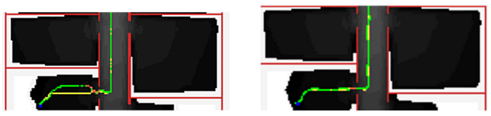

Figure 8: Movement without (left) and with (right) drive control

Second, we implemented a limitation in the speed increment from one schema iteration to the next, avoiding too sharp accelerations. As can be seen in figure 8, without this drive control (left) the robot tried to get the top speed in very short time and the inertia deviated the robot from the reference route. With drive control (right) the robot closely followed the reference route.

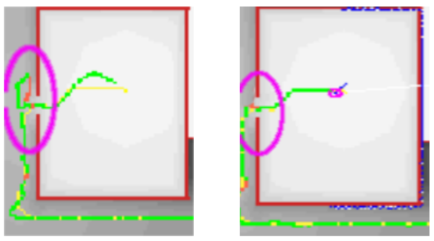
Figure 9: Movement without (left) and with (right) speed control

Third, a speed control slow down the robot when it gets close to final destination. The value of the field is a good approximation of the distance to goal. This speed control was inserted in the fuzzy controller of Follow Gradient schema. As can be seen in fig 9, with the speed control (right) the robot approaches properly to destination, avoiding the effects of the inertia like missing a door or zigzag at the end of the route (left).

5 Conclusions

An architecture for behavior generation named JDE has been presented. It proposes a collection of perceptive and actuation schemas to generate robot behavior. Such collection is organized as a hierarchy that can change its shape according to the situation. The action selection is decomposed into several small competitions, bounding its complexity.

The navigation behavior of an indoor robot has been designed inside JDE. It combines two well known techniques: the deliberative approach Gradient Path Planning and the reactive Virtual Field Force algorithm. Both have been implemented with JDE-schemas, and combined in a small hierarchy. The hierarchy reconfigures itself in the presence of close obstacles.

The robot exhibits good performance in the experiments: it reaches the remote target and properly avoids expected and unexpected obstacles, even dynamic ones.

We are currently working to probe the described navigation behavior in the real robot. We are also introducing new behaviors into the pool, to test JDE habilities to integrate a larger number of them.

References

[1] Arkin R.C., *Motor schema-based mobile robot navigation*, Int. Journal of Robotics Research 8(4):92-112, August 1989.

[2] Arkin R.C., Fujita M., Takagi T. and Hasegawa R., *An ethological and emotional basis human-robot interaction*, Robotics and Autonomous Systems 42:191-201, 2003.

[3] Brooks R., *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation, 2(1): 14-23, March 1986.

[4] Simmons R., Goodwin R., Zita K., Koening S. and O'Sullivan J., *A layered architecture for office delivery robots*, Proc. of the ACM Int. Conf. Autonomous Agents, pp 245-252, Feb 1997.

[5] Konolige K., *A gradient method for real time robot control*, Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS-2000), pp 639-646, Nov 2000.

[6] Borenstein J. and Koren Y., *The vector field histogram fast obstacle avoidance for mobile robots*, IEEE Journal of Robotics and Automation, 7(3):278-288, June 1991

[7] Cañas J.M. and Matellán V., *Dynamic schema hierarchies for an autonomous robot*, Lecture Notes in Artificial Intelligence vol 2527, pp 903-912, 2002.

[8] Konolige K. and Myers K.L., *The Saphira architecture for autonomous mobile robots*, Artificial Intelligence and Mobile Robots: case studies of successful robot systems, pp 211-142, MIT Press, 1998.