Behaviour Cooperation by Negation for Mobile Robots

Pedro F. Santana  
IntRoSys, S.A.  
Campus FCT-UNL, 2829-516 - Portugal  
pfs@uninova.pt

Luís Correia  
LabMAg - University of Lisbon  
Campino Grande, 1749-016 – Portugal  
Luis.Correia@di.fc.ul.pt

Abstract— This article presents a behavioural architecture, the Survival Kit (SK), which allows behaviours to cast their multi-valued output by means of constraints over an 'action feature space'. In other multi-valued based behavioural architectures, behaviours cast their output directly over a set of 'action spaces'. As the number of degrees of freedom increases, the amount of data flowing from behaviours to the coordination node, the required memory, and the computational demand to select the best action increase exponentially. By using an 'action feature space', the SK reduces greatly these problems. Furthermore, the 'action feature space' is endowed with inertia, which provides the architecture with a mechanism to automatically handle noisy sensors, to create Fixed Action Patterns, and to maintain local implicit representations of the environment. Special attention is given to an environmental feature, the gap, which allows to optimise paths based on immediate ranging data. Experimental results in a physical robot demonstrate the ability of the architecture to control robots with noisy sensors.

Index Terms—Mobile robots, Behaviour-Based Architectures, Sensor-Based Local Navigation, Survival Kit Architecture

I. INTRODUCTION

Service robots must be affordable and sustainable in order to become ubiquitous, requiring their control systems to: (1) be simple, (2) not rely in complex sensory apparatus, and (3) have good scalability. For this type of control, behaviour-based architectures are more convenient as they are intrinsically parallel and reactive.

In some behaviour-based architectures (e.g. [1], [2], [3]) each behaviour generates the best action in its perspective (i.e. a single-valued output). In other architectures (e.g. [4], [5]) a behaviour's output expresses how interesting each of the possible actions is (i.e. a multi-valued output). In the multi-valued case the information loss between behaviours and coordination node is smaller. As a result, the coordination node aggregates more information to better handle situations where different behaviours cast conflicting actions.

This paper proposes the Survival Kit (SK) architecture, which allows behaviours to cast constraints over an action feature space, instead of acting directly onto the action space. This type of cooperation by negation, where reflexes cooperate by adding constraints, guarantees that the resulting action does not disagree with any of the reflexes. Since reflexes can cooperate the resulting behaviour can remain goal-oriented without tedious parameter tuning. A constraint applied to an action feature implicitly represents a set of constraints applied to a sub-set of the action space. Hence, although the amount of data flowing towards the coordination node is smaller in the SK case, the amount of information embedded on it is much greater (i.e. there is a data compression effect). This also results in reduced memory requirements.

Previous work modelled the action space upon which behaviours cast their multi-valued preferences in different ways. In [4] behaviours can vote for actions in different action spaces (e.g. steering and speed). In [5] there is a single one encompassing all actuators. Although the latter allows the exploitation of coherence among the actions selected to each actuator (not necessarily the case of the former), its complexity grows exponentially with the number of degrees of freedom (i.e. it has no bounded complexity). Some heuristics can be employed to reduce the problem but conceptually it remains. This is in fact an obvious conclusion if one considers that the system starts to exhibit a centralised nature.

In the SK case, the coordination node exploits the semantics embodied in each action feature so as to generate the action to be actually performed. An objective function selects a sector of the environment to head to, and the remaining action features are used to build the actual action. Hence, there is no demanding search mechanism operating directly over the action space.

Finally, the action feature space is endowed with inertia, which provides the architecture with a mechanism to naturally handle noisy sensors, to create Fixed Action Patterns, and to maintain local implicit representations of the environment.

The implementation is a SK instance of a simple yet robust goal-oriented obstacle avoidance method, integrating dynamic constraints in a unstructured indoor environment. Taking into consideration the tradeoff between performance and low computational load, it will be shown to have advantages over the most popular methods (e.g. [6], [7], [8], [3]).

II. THE SURVIVAL KIT ARCHITECTURE

The SK architecture is intended to be used as the bottom layer of a robot control system, providing it with safe local navigation capabilities. Thus, everything required to maintain the survival (in terms of immediate reactions) of the robot should be implemented within the SK architecture. Relying on the SK to guarantee safe local navigation, upper decision/control layers are relieved of handling real-time unexpected events.

Figure 1 illustrates the main components of the SK architecture. Briefly, a set of reflexes inhibits some of the available action features according to sensory information.
A. The Action Feature Space

The core of the architecture is the action feature space, which describes indirectly all available actions to the robot. An example of an action feature is \( v_{\text{max}} \), which stands for the maximum linear velocity allowed for a given sector of the environment. The action feature space is composed of two sub-spaces: the space-variant and the space-invariant.

1) Space-Variant Action Feature Sub-Space: The space-variant action feature sub-space is sectoral (see figure 1). Each sector \( j \in J \) with an angular size \( \phi \) centred in the robot, where \( J \) is the set of all possible sectors, corresponds to a region in the environment. This design is motivated by the fact that most sensory information is radial. Thus, a similar representation facilitates further processing. A set of space-variant action feature descriptors \( F_{sv} \) must be defined every time the SK architecture is instantiated. In each sector, associated to each feature descriptor \( f_{sv} \in F_{sv} \), there are two slots, one for a constraint \( c \) on the respective action feature, and another one for its temporal validity \( \tau \).

Setting a constraint \( c \) (e.g. to \( 1 \text{ ms}^{-1} \)) and its corresponding \( \tau \) (e.g. to \( 100 \text{ ms} \)) associated to a feature descriptor \( f_{sv} \) (e.g. \( v_{\text{max}} \)), in a given sector \( j \) (e.g. 2), will affect the set of available actions for that sector. Thus, a constraint added by a reflex on a space-variant action feature has the following format: \( \text{cons}(j, f_{sv}, c, \tau) \) (e.g. \( \text{cons}(2, v_{\text{max}}, 1 \text{ ms}^{-1}, 100 \text{ ms}) \)).

A set of conditions must be met when accepting a new constraint. If a new constraint reduces the possible set of actions (e.g. if the feature is \( v_{\text{max}} \) with a current value of \( 1 \text{ ms}^{-1} \) and the new constraint intends to reduce the value to \( 0.5 \text{ ms}^{-1} \)), then it is immediately accepted. If the new constraint validity is greater than the current one, then the constraint validity is updated with the newer value.

The constraint validity can be used for three purposes: to reduce the system sensitivity to noisy sensors, to specify Fixed Action Patterns, and to create a myopic local environment representation in terms of space-variant action features. The representation is myopic due to the absence of self-localisation mechanisms, which, in general, hinders spatial transformations, specially translations. However, if heading information exists, the space-variant action feature sub-space can be geo-referenced. In such case, if the robot is pointing north and a constraint is set to sector \( j = 0 \), while still valid, that constraint will always refer to the sector pointing north independently of robot’s subsequent headings.

2) Space-Invariant Action Feature Sub-Space: A second action feature sub-space, the space-invariant action feature sub-space, allows reflexes to constrain certain dynamics of the robot, such as angular velocities, independently of any spatial relationship. Setting a constraint in this sub-space \( F_{si} \) is in everything similar to the previous case, except that a feature \( f_{si} \in F_{si} \) is considered instead, and any spatial relationship discarded. Thus, a constraint in this sub-space is defined by \( \text{cons}(f_{si}, c, \tau) \). Notice that \( F_{sv} \cap F_{si} = \emptyset \).

B. Reflex Pools

Two loops can be found in the SK architecture. A set of reflexes perceives the world, acts upon the action feature space (i.e. produces a set of constraints), an action is selected which will change the world, and in turn the world affects robot’s perception. This is the first loop. The second loop is internal, where reflexes are able to sense actions produced by other reflexes, a sort of internal stigmergy or blackboard.

To implement the second loop, reflexes are split into several pools, which are iterated in sequence. The system designer can allocate reflexes to pools as required. In general, reflexes of pool \( n \) will support their actions on information set by reflexes of pools \( m < n \), described in terms of constraints.

C. Descendant Pathways

Upper layers are allowed to modulate the SK, via two descendant pathways, one inhibitory and other excitatory. Via the inhibitory pathway, upper layers can constrain the action feature space before reflex pools. To increase plasticity, the inhibitory pathway also allows upper layers to suppress reflexes’ output.

In opposition to the inhibitory pathway, the excitatory one is exclusive; hence, in order to guarantee that only one behaviour can send a message to the SK through this channel, a coordination mechanism must be provided. The excitory modulation signal allows to select, configure, and request the coordination node for a certain motor action pattern. The SK architecture could include a coordination node allowing several behaviours to send excitatory signals, which would require to commit to a coordination philosophy. Since the goal of the SK architecture is confined to provide safe local navigation, such decision is left to upper layers.

D. Coordination Node

The SK can aggregate as many coordination nodes as desired; still, only one can be active at a time, which is selected...
and configured via the descendant pathway. Constrained by
the available action features, and modulated by the excitatory
signal, the coordination node creates an action according
to a given criteria. The motor actions producer module is
responsible for converting the created action into robot’s
actuators outputs, and to cooperate with the coordination node
so as to guarantee that a certain action is feasible in terms of
actuators as well as respects both action feature sub-spaces
constraints.

We define the concept of main axis of motion, which may
not coincide with the robot’s front. The main axis of motion
is displaced by an angle $\alpha$ from robot’s front, and represents
the side of the robot that should match the desired heading, i.e.
the desired posture of the robot. The coordination node can
dynamically change the main axis of motion if no action can
be found with the current one. Then, the previous iteration
state of the action feature space is restored, reflexes are
reiterated and a new action is produced.

III. A Survival Kit Instance

This section presents a SK instance for an indoor circular
robot with synchro-drive wheelbase. The robot carries a 12
elements full-sonar ring for obstacle detection, collisions
are detected by inspecting the current sunked by motors,
and heading for demonstration purposes is determined via
odometry.

A. Action Feature Space

$F_{sv} = \{v_{max}, a_{max}, enable\}$ defines the space-variant
action feature set, which represents for a given sector, the
maximum linear velocity allowed, the maximum distance free
of obstacles, and a flag indicating if movement in that sector
is allowed at all, respectively. When a feature is constrained,
with its current one. Then, the previous iteration
state of the action feature space is restored, reflexes are
reiterated and a new action is produced.

We define the concept of main axis of motion, which may
not coincide with the robot’s front. The main axis of motion
is displaced by an angle $\alpha$ from robot’s front, and represents
the side of the robot that should match the desired heading, i.e.
the desired posture of the robot. The coordination node can
dynamically change the main axis of motion if no action can
be found with the current one. Then, the previous iteration
state of the action feature space is restored, reflexes are
reiterated and a new action is produced.

### A. Action Feature Space

$F_{sv} = \{v_{max}, a_{max}, enable\}$ defines the space-variant
action feature set, which represents for a given sector, the
maximum linear velocity allowed, the maximum distance free
of obstacles, and a flag indicating if movement in that sector
is allowed at all, respectively. When a feature is constrained,
with its current one. Then, the previous iteration
state of the action feature space is restored, reflexes are
reiterated and a new action is produced.

We define the concept of main axis of motion, which may
not coincide with the robot’s front. The main axis of motion
is displaced by an angle $\alpha$ from robot’s front, and represents
the side of the robot that should match the desired heading, i.e.
the desired posture of the robot. The coordination node can
dynamically change the main axis of motion if no action can
be found with the current one. Then, the previous iteration
state of the action feature space is restored, reflexes are
reiterated and a new action is produced.

1) Range-Based Reflex: In the experiments performed for
this work, the robot uses a full-sonar-ring of 12 elements,
which are displaced by $\alpha = \frac{2\pi}{12}$ radians. An obstacle $i$
has the following format, $(\beta_i, r_i)$, where $\beta_i$ is the angle between
the front of the robot and the obstacle’s direction, and $r_i$
is the distance to the obstacle.

In the following text, the term obstacle will be used to
something that is perceived by a sonar, and not necessarily to
a complete object in the environment (see figure 2(a)); that
is to say that there is an obstacle per each sonar that detects
something. This approach avoids the computational burden
of creating elaborate representations of obstacle shapes as
they are in the environment. The $d_{max}$ feature of each sector
containing an obstacle $i$, is constrained by the range data $r_i$.

It is necessary to enlarge the obstacle so a sector not
containing it is in fact navigable; i.e. obstacles have to be
considered in the configuration space (c-space). This work
simplifies this problem by exploiting the fact that obstacles’
angular size is equal to the sonar’s field of view (i.e. it
is fixed). Hence, taking into account the ratio between ob-
stacle’s diameter, roughly defined as $d_i = r_i \cdot \sin \alpha$, and
robot’s diameter $w$, it is possible to determine, in the c-
space, all sectors affected by obstacle $i$; namely, $j_{obs_i} \in \Omega(\Delta_i - \beta_i), \Omega(\Delta_i + \beta_i)$, where $\Omega(.)$ returns the sector of
a given angle, and $\Delta_i = \frac{1}{2} \cdot \alpha \cdot \left(\frac{w}{r_i \sin \alpha} + 1\right)$. Note that
no trigonometric function is computed in runtime ($\alpha$ is a constant)
so as to reduce the computational requirements.

Then, the $d_{max}$ feature of each sector affected by the
enlarged version of the obstacle is constrained by adding, for
each sector $j_{obs_i}$, a constraint $cons(j_{obs_i}, d_{max}, r_i, 1000 \text{ ms})$
(constraint validity values have been set empirically for all
reflexes). The process is repeated for all detected obstacles.

2) Touch Reflex: This reflex intends to handle situations
where the robot collides with obstacles, which by their nature,
e.g. small height, are not detectable by the sonar ring. When
the current sunked by the robot’s motors goes above a certain
threshold a collision is detected. This reflex reports the ob-
stacle by adding a constraint $cons(j_{touch}, d_{max}, 0, 7000 \text{ ms})$
per each $j_{touch} \in \Omega(-\frac{\pi}{2}, \Omega(\frac{\pi}{2}))$. To prevent the robot
rotating as it moves away from the obstacle, constraints
$cons(\omega^+, false, 3000 \text{ ms})$ and $cons(\omega^-, false, 3000 \text{ ms})$ are
also set. Finally, as to guarantee that the robot does not choose to stop the constraint $cons(stop, false, 3000 \text{ ms})$
is added. The result is that any movement along the main axis
of motion is disabled and there is a stop = false constraint
to satisfy. As a consequence there is no available action. Yet,
as it will be detailed in section III-D, in such a situation the
system reconfigures itself to search for other solutions, such
as moving backwards. Briefly, after the collision, the robot
moves straight backwards for three seconds. Afterwards, it
turns away from the detected obstacle bearing towards best
direction; progressively the robot gains velocity. Meanwhile
the $d_{max}$ constraint validity expires and the robot moves as
that obstacle is not there anymore. Thus, this reflex triggers a
Fixed Action Pattern described in terms of validity constraints.
that helps the robot to contour an obstacle invisible to the range sensors.

3) Dynamic Constraints Reflex: The maximum linear velocity allowed for each sector \( j \), considering a linear trajectory and uniformly accelerated movement, is
\[
v_{\text{allowed}}(j) = \sqrt{2 \cdot v_{\max} \cdot v_{\min}(j)} - d_{\text{safety}},
\]
where \( v_{\max} \) is the maximum deceleration capability of the robot, \( r_j \) is the minimum linear distance free of obstacles in the respective sector, \( d_{\text{safety}} \) is the minimum distance the robot is allowed to be from an obstacle, and \( v_{\min} \) guarantees that \( r_j - d_{\text{safety}} \) is never negative. Thus, the maximum velocity allowed is such that guarantees the robot does not hit any obstacle when decelerating as much as it can. This acceleration can be set for other purposes in addition to dynamic constraints; for instance, to guarantee that robot’s payload does not suffer accelerations and vibrations greater than certain values. Then, a constraint \( v_{\min}(j) = \max(\lambda_1 \cdot r_d(C_j), v_{\min}(j)) \) is added per each sector, where \( v_{\min} \) is the minimum velocity of all \( v_{\text{allowed}} \) calculated in sectors between the main axis of motion and \( j \).

Since this reflex is in the second pool, it can measure the free distance to obstacles by perceiving the \( d_{\text{max}} \) action feature of the corresponding sector (i.e. to make \( r_j = d_{\text{max}} \) itself and the obstacle. Finally, \( r_d(C_j) \) is the largest of all \( r_d(C_i) \).

Then, considering robot’s dynamic constraints, this reflex disables those sectors that would require unfeasible actions. In particular, a constraint
\[
\text{cons}(j_{\text{dyn}}, \text{enable}, \text{false}, 1 \text{ iteration})
\]
is added per each sector \( j_{\text{dyn}} \), \( j_{\text{dyn}} \) is a sector whose \( v_{\max} \) is smaller than the robot’s minimum velocity achievable until the next iteration (i.e. taking into consideration \( a_{\max} \)).

C. Descendent Pathways

This instance is limited to the use of the excitatory descendent pathway, through which the upper-layer provides both desired linear velocity and desired direction of motion, as well as the maximum range to consider obstacles, i.e. \( m \). The simplicity of the upper-layer discards the need for an inhibitory descendent pathway.

D. Coordination Node

The coordination node computes the action to be performed by respecting the constraints set onto the action feature space and by taking into account the modulatory signal provided by the upper-layer. When integrating the constraints associated to adjacent sectors, the coordination node recurs to the environmental feature \( \text{gap} \) to assess the free space connectivity between them. Theoretically a \( \text{gap} \) represents a discontinuity greater than the width, \( w \), of the robot in the \( d_{\max} \) feature between two adjacent sectors. In practice, this value may be magnified by a factor \( f \) in order to compensate, to some extent, the low angular resolution of the range sensor set, which may introduce some fake \( \text{gap} \)s. From the two sectors involved in the discontinuity, the one with greater \( d_{\max} \) is considered to be a \( \text{corridor} \). Since a \( \text{corridor} \) is defined in the \( \text{c-space} \) (by means of \( d_{\max} \)), the robot is kept safe of collisions as it moves in it (see figure 2(b)). Every sector in which the \( d_{\max} \) feature is greater or equal to \( m \) (a value provided in the excitatory signal) is also considered to be a \( \text{corridor} \). Reducing \( m \) allows the robot to move towards obstacles and consequently approaching a goal set between itself and the obstacle. Finally, \( d_{\max} = \min(m, d_{\max}) \).

The action to be actually effected by the robot in iteration \( n \) is composed of a selected sector, \( s \), and a selected linear velocity. If the maximum linear velocity allowed in the selected sector is smaller or equal than the desired linear velocity, provided in the excitatory signal, then it is selected; otherwise, the desired linear velocity is the one selected.

First, all sectors inhibited by other reflexes via the \( \text{enable} \) action feature are removed from further computations. Then, the \( \text{best corridor, } b_{c[n]} \), taking into account the information present in the space variant action feature sub-space is chosen by maximising the following objective function:
\[
b_{c[n]} = \max_i (\lambda_1 \cdot r_d(corr_i) + \lambda_2 \cdot \text{dist}_{\perp}(corr_i, s[n-1]))
\]
where, the first term \( (r_d) \) provides goal-oriented behaviour, the second term \( (\text{dist}_{\perp}) \) allows to commit to the previously selected sector in order to reduce oscillatory behaviour (similar to the definition provided in [7]), and weights \( \lambda_1 \) and \( \lambda_2 \) enable a proper integration of both components, all normalised between \([0,1]\).

\( r_d(corr_i) \) refers to the \( d_{\max} \) value of \( corr_i \) projected on the desired direction axis (see figure 2(b)). \( \text{dist}_{\perp} \) grows with the
angular distance between $geo_L(\text{corr}_i)$ and $geo_{L}[n-1](s_n[n-1])$, with $geo_L(\cdot)$ transforming the bisector angle of a given sector to geo-referenced coordinates, whereas $geo_L[n-1]$ does the same for iteration $n-1$. This mechanism is simple to tune due to the little number of parameters, which only exist to reduce possible oscillations.

In order to remove heading static errors, $\lambda_2$ is made dynamic. Briefly, if the robot’s heading is nearby the desired heading, and the produced action does not change significantly between consecutive iterations (i.e. $(\text{heading error} < \frac{\pi}{8}) \land (\mathcal{L}(geo_L(s_n[n]), geo_{L}[n-1](s_n[n-1])) < \epsilon)$), where $\mathcal{L}$ returns the angular distance between two given angles), then $\lambda_2$ is decreased at rate $\tau_{\lambda_2}$; otherwise, $\lambda_2$ is set to its maximum value, $\epsilon$. As a result, in the presence of heading static error the $\text{dist}_L$ component is reduced and the robot becomes more goal-oriented. If instead a sudden change of heading is required (e.g. due to the presence of an obstacle) the former weights trade-off is restored.

Now that the best corridor has been determined by taking into account space variant information, it is necessary to determine the selected sector $s_n$, which has to cope with the space invariant constraints as well. This is done by asking the motor actions producer to filter all unfeasible sectors (see section II-D). The feasible sector whose angular distance to $b_c[n]$ is the smallest is $s_n[n]$.

If moving forward (i.e. main axis of motion set to 0) is unfeasible, then moving backwards is tried (i.e. main axis of motion set to $\pi$). Since the robot can not travel sideways, no other main axis of motion could be selected. After the change, reflexes are reiterated and the coordination node process repeated. This is in fact what happens in the touch-reflex, where no motion is possible if moving forward, which forces to try moving backwards.

As aforementioned, the robot has a synchro-drive wheelbase, i.e. an action is composed of a linear velocity and an angular velocity. The linear velocity is set by the coordination node as previously described, and the angular velocity is proportional to the angle between the selected sector and the robot’s front. This process is implemented within the motor action producer, which also confirms the feasibility of an action, taking into consideration the locomotion method.

IV. EXPERIMENTAL RESULTS

For the experimental setup we have used a B12 Robot from Real World Interface Inc., which implements the SK instance previously presented. In the following experiments the SK is modulated for the robot to move “North” (towards the top in figure 4) at a speed of 0.3 m/s. $\lambda_1$, $\epsilon$, $a_{max}$, $m$, $\phi$, and $f$, and $\tau_{\lambda_2}$ have been set to 1.0, 0.5, 0.2 m/s, 5 m, 0.049 rad (i.e. 128 sectors), 1, and 1 s respectively.

In order to cope with some sensor limitations, further parametrisations have been made. First, sensors get saturated with ranges smaller than 0.1 m. Secondly, the low sampling rate of sonars, 2 Hz, affects the maximum allowed speed. To take this into consideration one could search for the maximum speed at which the robot could travel during the current iteration that is low enough to guarantee a safe stop (i.e. braking as much as it can) before the closest obstacle during the subsequent iterations (i.e. similar to [6]). To avoid such a search (processing consuming) method, $d_{\text{safty}}$ is increased with the maximum distance the robot can travel between two iterations, i.e. $d_{\text{safty}} = 0.1 \text{ m} + (0.3 \text{ m/s} \times 0.5 \text{ s}) = 0.25 \text{ m}$. As a result, the robot will approach obstacles according to its “real” width and then will be stalled because $d_{\text{safty}}$ is ”artificially” high. So, the width of the robot is increased in 0.1 m, i.e. $w = 0.4$ m for a proper sector selection.

![Fig. 3. Experimental setup.](image)

Figure 3 shows the environment where the experiments took place. At the start of each run the robot is placed as illustrated in figure 3(a). The robot performs consistently well, proceeding in a smooth way negotiating obstacles and possible collisions. The results of two typical runs in the environment in question are presented in figures 4(a) and 4(b) respectively. The red circles represent the robot’s contour and the green ‘x’ represents all sonar hits along the run. This data is traced based on odometry. Black lines represent, roughly, ranging data of particular corridors.

In the first run, the robot chooses to move towards the left gap since it is the one embodying more potential for progression towards “North”, in the robot’s perspective. As a result the robot moves along the gap until another one emerges at the robot’s right side. Then the robot selects the latter gap and avoids the u-shaped obstacle. Therefore, the robot follows the shortest path based on immediate ranging data, in a very smooth way as well as keeping the requested speed whenever it is possible (i.e. most of the time).

![Fig. 4. Experimental results of two runs (representation to scale with a 2m x 2m grid superimposed).](image)
In a second run, a person appears at the left of the first set of obstacles (see the grey circle in figure 4(b)). This new obstacle changes the local perspective of the free space. In particular, the left gap is no longer more interesting than the right one. Due to odometry errors it is not possible to see it very clearly in the figure, where the set of obstacles in the front of the robot at the beginning of the run are erroneously slightly bended to the right. The commit component is responsible for suppressing oscillations between the two gaps, which are very similar in terms of range projected over the desired direction. The arrows in figures 3(b) and 4(b) are signalling a collision, which could not be avoided for the smooth surface of the obstacle in question could not be perceived at all (there was no other collision in both runs). As a consequence of the collision, the touch reflex induces the robot to back off from the obstacle for a while and then to turn rightwards. In this second run the speed changes more often due to the more difficult trajectory. It should be taken into account that the type of obstacles, the enormous amount of noisy readings and the clutterness of the environment make it no easy job.

V. DISCUSSION

To our knowledge, only the works presented in [9], [10], [2] have explored the concept of dynamical components (i.e. with memory) within their architectures. [9] and [2] explored dynamics for behaviour arbitration whereas [10] considered dynamics of the variables themselves. Hence, the SK dynamical component action feature space, which is implemented in terms of the action features constraint validity, is more in line with the work of [10] than with the other two. However, the SK does not accept changes in its internal variables without considering if they reduce the robot’s safety, which is not the case in the architecture presented in [10].

It was demonstrated that the way gaps have been exploited in the SK instance resulted in a solution exhibiting a stable, safe, and goal-oriented behaviour. This was only possible because the coordination node is an aggregation unit. In [11], the gap environmental feature is used just as a mean of producing a topological map. In [8] the same feature is also used but not as a direct target of motion.

The employed objective function has some resemblances to those proposed by some obstacle avoidance methods. However, since $r_d$ already embodies information on distance to obstacles, free-space connectivity, heading error, and speed, i.e. a sort of terrain progression towards the goal metric, the objective function is simpler and more predictable than those used in [6], [7], where all those features have to be considered explicitly.

VI. CONCLUSIONS AND FUTURE WORK

Experimental results show that the concept of ‘cooperation by negation over a set of action features’ aggregates all the ingredients necessary to implement scalable multi-valued based behavioural architectures. These ingredients include low memory and low bandwidth requirements, as well as bounded complexity. By endowing action features with inertia, it became possible to handle noisy sensors as well as to implement Fixed Action Patterns with stateless behaviours. A specific implementation showed the ability to implement a robust, local minima free, and predictable obstacle avoidance method considering robot’s dynamic constraints.

Despite the significant enhancements SK brought to the concept of multi-valued behaviour’s output, the coordination node and the action feature space are still centralised entities, being in the limit a bottleneck. Only competitive-based behavioural architectures can claim to be fully distributed with full bounded complexity. This suggests that the SK should evolve in the direction of becoming fully distributed, which is a challenging task if the modularity and smoothness of cooperative approaches are to be kept intact. All synchronisation signals (e.g. reflex pools sequential iterations) should also be removed so as to attain true scalability. It will also be investigated the possibility of layering the SK so as to use the same concepts to manage higher level actions (e.g. GO/HOME). The generalisation should be possible as the system is not restricted to handle actions at the actuator level.

REFERENCES