Generating Interactive Robot Behavior: A Mathematical Approach

Rainer Menzner  Axel Steinhage  Wolfram Erlhagen
Institut für Neuroinformatik, Lehrstuhl für Theoretische Biologie
Ruhr-Universität Bochum, 44780 Bochum, Germany
Tel.: +49 234 322 5564, Fax.: +49 234 321 4209
Menzner/Steinhage/Erlhagen@neuroinformatik.ruhr-uni-bochum.de

Abstract

This paper describes a generalized mathematical approach to generate and organize robot behavior. For the example of a simulated robot which can perform a number of elementary behaviors we demonstrate how the internal behavioral state of the system can be influenced by external inputs. The inputs consist of keyword command strings given by an operator and of depth information determined by the simulation environment. The robot’s behavioral state is defined by the activity pattern of the set of elementary behaviors and by the values of the continuous variables that control these elementary behaviors. Sensor information acts on the system by specifically stable states for the dynamics. Behavioral stability is guaranteed by a number of mathematically well defined design principles which must be obeyed when designing behavior generating systems following our approach. In this paper we describe the characteristics of this approach before we present the simulation of a robotic system which displays complex behavior.

1. Introduction

Both, biological and artificial behavior generating systems must solve the following two basic problems: 1) They must control their effectors depending on the information obtained from the external world by means of their sensors and 2) they must organize the elements of their behavioral repertoire such that goal directed behavior is produced.

The first problem describes the necessity to react according to the environment on a short time scale. This low level control of the effectors generates reactive behavior which we call elementary behavior as it mostly shows up in a very stereotypic behavioral pattern. For biological organisms behaviors like fleeing, eating and catching belong to this class, for artificial systems, i.e., robots, we speak of obstacle avoidance, target acquisition etc. The main requirement when generating these behaviors is the stability of the underlying control system and the main problem is the correct interpretation of the sensor information. As the entire sensor input influences the control system continuously while the behavior is acted out, we call the development of the behavior over time a parametric change of the behavior.

The second challenge for behavior generating systems is the organization of the behavior. This basically requires to activate and deactivate elementary behaviors in a temporally and logically ordered fashion. As an example, consider the behavioral goal “go left to reach the main station”. This goal is formulated on an abstract level which means that it resembles more a plan than a direct command: if the behaving system is in the middle of a street, it would at first have to go straight to the next crossing and then turn left. Abstractly spoken, the system has to organize the elementary behaviors straight movement, obstacle avoidance and turn left according to the current situation such that the overall goal is reached. Our simple example could likewise be solved by blending the elementary behaviors obstacle avoidance and target acquisition (Bicho and Schöner, 1997) (by considering the street’s borders as obstacles), however, there are more complex situations, in which the activation and deactivation of the elementary behaviors is triggered by a sequence of sensor signals. In contrast to the parametric change, we call these triggered changes of the behavior a categorical change or switching. Of course, the decision which elementary behaviors to activate and which to deactivate does not only depend on the presence of the trigger signal but also on the pattern of currently active elementary behaviors itself. If, for instance, an elementary behavior has high priority (such as fleeing or obstacle avoidance), it may prevent the activation of another elementary behavior with lower priority. Conversely, the activation of an elementary behavior may require the simultaneous activity of other elementary behaviors.

A system for generating and organizing behavior must account for all these aspects. So far, there exist only partial solutions which either deal with the generation

---

Footnote: For example in biology there exists the “egg roll” effect, where the visual trigger signal of a spherical object outside the nest activates the behavior of a grey goose to roll the object “back” to the nest (Gould and Grant Gould, 1994)
of certain elementary behaviors such as target acquisition, obstacle avoidance, wall following etc., or which describe architectures for behavioral organization (see e.g., Maes, 1988)). In the following we will present a unified mathematical approach which applies to the control of elementary behaviors as well as to behavioral organization. In the following section, we will briefly describe the basic mathematical elements of this approach before we present a robot simulation system which interacts with an operator by using this “toolbox” to generate and organize its behavior. A more detailed description is contained in (Schöner et al., 1995) and (Steinhage, 1998).

2. The Dynamic Approach to Robot Behavior Generation and Organization

The basic idea of the approach is to parameterize an elementary behavior \( i \) by a scalar or vector valued behavioral variable \( \mathbf{x}_i \in \mathbb{R}^N \). For example, locomotion of a robot in 2D can be expressed by the vector \( \mathbf{x}_{\text{loc}} = (v, \phi) \) of the robot’s forward velocity \( v \), its heading direction \( \phi \) and a variable \( [0,1] \) which indicates to which degree the locomotion behavior is active. Behavior is generated by assigning values to \( \mathbf{x}_i \) which are used by the motor system of the robot to drive the wheels or joints of the robot’s effectors.

Now comes the special feature of our approach: we design the differential equation of a dynamical system

\[
\tau_i \dot{\mathbf{u}}_i(\mathbf{x}_i, t) = \mathbf{F}_i(\mathbf{u}_i(\mathbf{x}_i, t), \mathbf{s}_i(\mathbf{x}_i, t))
\]

(1)

the solution of which generates the desired behavior \( i \).

In (1), \( \mathbf{u}_i(\mathbf{x}_i, t) \) is a function of the behavioral variable, \( t \) is the time, \( \dot{\mathbf{u}}_i(\mathbf{x}_i, t) = \frac{\partial \mathbf{u}_i(\mathbf{x}_i, t)}{\partial t} \) is the derivative of \( \mathbf{u}_i \) with respect to time, \( \mathbf{s}_i(\mathbf{x}_i, t) \) is a vector of parameters, the so-called input and \( \tau_i \) is the time scale of the dynamics. For the function \( \mathbf{u}_i(\mathbf{x}_i, t) \) our approach offers two choices so far: In the first case, \( \mathbf{u}_i \) is the excitation of a so-called neural field defined over the behavioral variable \( \mathbf{x}_i \). The activity of the field represents to which extent the corresponding value of the behavioral variable is specified. We will give an example in section 3.5. In the second, so-called instantiated case, we set \( \mathbf{u}_i(\mathbf{x}_i, t) \equiv \mathbf{x}_i(t) \) such that the dynamics (1) simplifies to

\[
\tau_i \dot{\mathbf{x}}_i(t) = \mathbf{F}_i(\mathbf{x}_i(t), \mathbf{s}_i(\mathbf{x}_i, t))
\]

(2)

The concrete form of \( \mathbf{F}_i \) depends on the behavior to generate and is subject to the designer’s skill. However, our approach contains some general rules which facilitate the design process and help to keep the dynamics stable: First, the dynamics must be designed such that desired values \( \mathbf{x}_i^d \) of the behavioral variable specify stable states and undesired values \( \mathbf{x}_i^u \) specify unstable states of the dynamics (2). This requirement ensures that the dynamics will keep the behavioral variables near desired values even when noise or perturbations act on the system. The values \( \mathbf{x}_i^d \) of the stable states are specified by the system’s sensors by means of the input vector \( \mathbf{s}_i(\mathbf{x}_i, t) \). For example, for target acquisition the system’s sensors measure the angular position \( \phi_{\text{tar}}(t) = s_{\text{loc}}(t) \) of a target. Then, a dynamics for the \( \phi \)-coordinate of the behavioral variable \( \mathbf{x}_{\text{loc}} \) could be:

\[
\tau_{\text{loc}} \dot{\phi}(t) = \phi_{\text{tar}}(t) - \phi(t).
\]

This dynamics has a stable fixed point (attractor) at \( \phi = \phi_{\text{tar}} \) and therefore the behavioral variable (and hence the robot’s heading) will relax to this value. Multiple desired behaviors can be accounted for by designing a dynamics with multiple stable states such that phase changes or bifurcations of the dynamics result in changes of the behavior. Undesired states, such as the direction towards obstacles, can be accounted for by setting unstable fixed points (repellers) for these values (Steinhage and Schöner, 1997).

The second rule of our approach states that the time scale on which the input \( \mathbf{s}(\mathbf{x}_i, t) \) changes must be much slower than the time scale \( \tau_i \) of the dynamics (see (Steinhage and Schöner, 1997) for a discussion). This is obvious for the example of target acquisition: if the angular position of the target changes faster than the dynamics can relax to the stable state, then the system does not work properly anymore. However, as the choice of the time scale \( \tau_i \) of the dynamics is up to the designer, the time scale requirement can always be met. This so-called separation of time scales also allows to couple multiple dynamics in the way that one slow dynamics \( j \) specifies the parameters \( \mathbf{s}_j(\mathbf{x}_j, t) \) of a second, faster dynamics \( i \). We make extensive use of this principle in section 3.3.

For the behavioral organization as discussed in the introduction, we have added a specific dynamics, a so-called arbitration dynamics, for the activities \( n_i \) of the elementary behaviors to our “dynamic toolbox”. This dynamics has stable fixed points at \( n_i = 0 \) (not active) and \( n_i = 1 \) (active) as will be presented in section 3.4 in more detail. The basic idea is to couple the \( n_i \)-dynamics of all elementary behaviors by means of fixed interaction matrices which code the pairwise logical relations between the behaviors. For this purpose we have developed a generalization of the well known BOOLEAN binary logic which takes dynamic variables in the range between zero and one as arguments. This will also be presented in section 3.4. For the specific application presented in this paper, the input \( \mathbf{s}(\mathbf{x}_i, t) \) of an \( n_i \)-dynamics consists of the activity of the other elementary behaviors \( n_{\text{not }}i \) and of a sensor signal from a keyword-recognizer which detects commands of an operator who advises the system to activate specific behaviors. In section 3.7 we will show how a simulated robot can be guided by these speech commands and how the problem of contradictions between the operator’s commands and the robot’s perceived depth information is solved. For the behavior
generating system of this robot we apply three types of
dynamics: 1) an instantiated dynamics of the form (2)
controls the movements of the robot platform in 2D, 2) a
neural field dynamics of the form (1) controls the point-
ing direction of the robot’s arm and 3) an instantiated
arbitration dynamics for the activities \( n_i \) of the ele-
mentary behaviors performs the behavioral organization.

In the following sections we will present a sample ap-
pllication which makes use of all the tools presented so
far. Our aim is to prove that this very limited number of
concepts allows for an implementation of a system which
generates complex behavior.

3. An Application: A Robot Simulation

3.1 Experimental Setup

In this section we will present a real application illus-
trating the topics discussed before. Figure 1 shows our
setup. The 2D scene consists of an arena in which the
robot, called JAVA\_ROBOT can move. The robot itself is
indicated by a square with an enclosed isosceles triangle,
determining the heading direction. The installed arm al-
 lows for pointing into directions of up to \( \pm 90^\circ \) from
the current heading direction. JAVA\_ROBOT is equipped with
two sensorial modalities. Firstly, a “vision” sensor pro-
vides a depth profile, ranging from \(-90^\circ\) to \(+90^\circ\) with
respect to the current heading direction. Furthermore,
the robot is sensitive to keyword speech. In this research
we use the small vocabulary given in Table 1. The recep-
tion of each of that words corresponds to the fullfil of
the so-called Sensor Context, \( s_i = 1 \), of the correspond-
ning behavior \( i \). There are also objects and obstacles in-
dicated by rectangles filled in dark gray color. Objects
and obstacles are not treated differently, their role will
become implicitly clear from the context. The area be-
low the arena in Figure 1 displays the currently perceived
depth profile, in which the pointing direction is marked
by a black rule, as well as the current position data.
The reference direction 0° in world coordinates is from
top to down in the arena and counts positive according to
mathematical conventions, i.e. counter clockwise.

On the steering level, the robot described above is
controlled by three parameters: (1) a heading velocity,
\( v_h \), (2) an angular velocity, \( \omega_h \), and (3) an arm pointing
angle, \( \phi_h \). Consequently, in order to control that robot,
a system of dynamics has to be set up which outputs
these three quantities and which allows to fuse depth
information from the vision sensor and keyword speech.

3.2 Aim of the Experiments

The aim of our investigations is to show that, using our
mathematical approach, it is possible to design robot
behavior that has a minimum built-in intelligence, pro-
vides a robot with the ability of “natural communication”
and combines all that with dynamical systems’ properties
like, e.g., stability. The target behavior of JAVA\_ROBOT is summarized as follows. Initially, that is,
without speech input, the robot does nothing. Upon
recognition of one of the action specification words, the
robot should start the corresponding action. “Moving”
should include simultaneous “Turning” but an explicitly
commanded “Turn” means “Turn on the spot”, that is,
it precludes “Moving”. “Pointing” stops both “Moving”
and “Turning” as it is vice versa. All actions may be
parameterized using the direction specifications. In ad-
dition to this, movement and turning may be accelerated
and decelerated. Whatever the robot does is related to
its physical environment. This means the robot would
try to avoid moving against a wall or an obstacle, even
if the blind execution of a speech command would lead
to such a collision. With respect to pointing this means
that the robot attempts to relate pointing direction spec-
fications to objects as long as there are any objects. This
topic may be termed “sensor fusion”.

In the following, we will first introduce the main dy-
namics that generates the values for \( v_h, \omega_h \) and \( \phi_h \), and
then describe how each of its components is extracted from the environment and processed by further dynamics.

### 3.3 Main Behavioral Dynamics

The main behavioral dynamics are based on the unified form of a sum of linear attractors, i.e.,

$$
\hat{\phi} = \sum_{i=1}^{N} n_i^2 \lambda_i (\phi_i - \phi), \quad n_i^2 \in [0, 1], \quad \lambda_i > 0.
$$

The arbitration scheme described earlier ensures that only one of the factors $n_i^2$ equals 1 so that the resulting attractor of (3) is just that corresponding $\phi_i$. The dynamical variable $\phi$ exponentially relaxes into this attractor $\phi_i$, whereby the time scale is given by the local $\tau_i = 1/\lambda_i$.

A second principle we make use of is time scale separation. In an example of two coupled dynamics

$$
\psi = \lambda_0 (\phi - \psi), \quad \phi \text{ as in (3)},
$$

$\psi$ will be able to follow its attractor $\phi$ stably as long as $\phi$ does not vary too fast, or, put another way, $\lambda_0 \gg \lambda_0$. The concept of coupling linear attractor dynamics on separate time scales is also discussed in more detail in (Menzer and Steinhauser, 1999).

After having discussed the underlying principles, we start the description of our system by examining the set of dynamics for the heading velocity:

$$
\nu_{\text{abs}} = \begin{cases} n_{\text{faster}}^2 \lambda_{\text{faster}} & (V_{\text{max}} - \nu_{\text{abs}}) \\ + n_{\text{slower}}^2 \lambda_{\text{slower}} & (V_{\text{min}} - \nu_{\text{abs}}) \end{cases}
$$

$$
\nu_{\text{dir}} = \begin{cases} n_{\text{mforward}}^2 \lambda_{\text{mforward}} & (\nu_{\text{abs}} - \nu_{\text{dir}}) \\ + n_{\text{mbackward}}^2 \lambda_{\text{mbackward}} & (-\nu_{\text{abs}} - \nu_{\text{dir}}) \\ + n_{\text{mstop}}^2 \lambda_{\text{mstop}} & (\nu_{\text{abs}} - \nu_{\text{dir}}) \\ + n_{\text{obst}}^2 \lambda_{\text{obst},\nu} & (\nu_{\text{obst}} - \nu_{\text{dir}}) \end{cases}
$$

$$
\nu_{\text{h}} = \begin{cases} n_{\text{m}}^2 \lambda_{\text{m}} & (\nu_{\text{dir}} - \nu_{\text{h}}) \\ + n_{\text{mstop}}^2 \lambda_{\text{mstop}} & (0 - \nu_{\text{h}}) \end{cases}
$$

Obviously, (5) complies to the format given by (3). All variables $n_i$, are activity values which stem from the nonlinear arbitrator dynamics which we will describe in section 3.4. The desired attractor is chosen by means of these values. The time scales grow faster from top to down: $\lambda_{\text{faster}}, \lambda_{\text{slower}} \ll \lambda_{\text{mforward}}, \lambda_{\text{mbackward}}, \lambda_{\text{obst},\nu} \ll \lambda_{\text{m}}, \lambda_{\text{mstop}}$. On the slowest timescale, $\nu_{\text{abs}}$ may vary between the fixed values $V_{\text{min}}$ and $V_{\text{max}}$. This is under direct control of the speech words slower and faster. If neither of these words has been encountered, we have $\nu_{\text{abs}} = 0$ which means $\nu_{\text{abs}}$ simply does not vary. This variable $\nu_{\text{abs}}$ is coupled as an attractor into the dynamics for $\nu_{\text{dir}}$ which runs on a faster time scale. Thus, $\nu_{\text{dir}}$ is guaranteed to relax on a stable trajectory into the possibly moving attractor $\nu_{\text{abs}}$. In case $n_{\text{mbackward}}^2 = 1$, the sign of $\nu_{\text{dir}}$ is inverted which then leads to a movement into backward direction. The third attractor restores the current velocity in forward direction if $n_{\text{mstop}}^2 = 1$, i.e., if movement explicitly or implicitly has been stopped. $\lambda_{\text{obst},\nu}$, usually being 0, becomes positive when the robot approaches obstacles. In that case in additional attractor at $\nu_{\text{obst}}$ arises and attractor averaging takes place. Subsection 3.6 discusses how $\lambda_{\text{obst},\nu}$ and $n_{\text{obst}}^2$ depend on the current object distribution. Finally, the $\nu_{\text{h}}$ dynamics on an even faster time scale is used to produce an output value $\nu_{\text{h}}$ that either follows the value of $\nu_{\text{dir}}$ (if moving is active) or attracts $\nu_{\text{h}}$ to 0 (in case moving is not active). A close look at (5) shows that the principle of time scale separation is violated for the case $n_{\text{mstop}}^2 = 1$ because we then have the same local time scale in both the $\nu_{\text{h}}$ and the $\nu_{\text{dir}}$ dynamics. This is, however, not problematic in this case since if $n_{\text{mstop}}^2 = 1$, the arbitration dynamics ensures that $n_{\text{m}}^2 = 0$ and hence $\nu_{\text{dir}}$ is not an attractor in the $\nu_{\text{h}}$ dynamics.

The set of dynamics producing a value for the head angular velocity, $\omega_{\text{h}}$, is only slightly more complicated:

$$
\omega_{\text{abs}} = \begin{cases} n_{\text{faster}}^2 \lambda_{\text{faster}} & (0 - \omega_{\text{abs}}) \\ + n_{\text{slower}}^2 \lambda_{\text{slower}} & (\Omega_{\text{max}} - \omega_{\text{abs}}) \end{cases}
$$

$$
\omega_{\text{int}} = \begin{cases} (1 - n_{\text{mbackward}}^2) \lambda_{\text{fip}} & (\omega_{\text{abs}} - \omega_{\text{int}}) \\ + n_{\text{mbackward}}^2 \lambda_{\text{fip}} & (\omega_{\text{abs}} - \omega_{\text{int}}) \end{cases}
$$

$$
\omega_{\text{dir}} = \begin{cases} n_{\text{leftfipsleft}}^2 \lambda_{\text{left}} & (\omega_{\text{left}} - \omega_{\text{dir}}) \\ + n_{\text{rightfipsright}}^2 \lambda_{\text{right}} & (\omega_{\text{right}} - \omega_{\text{dir}}) \\ + n_{\text{forward}}^2 \lambda_{\text{forward}} & (0 - \omega_{\text{dir}}) \\ + n_{\text{backward}}^2 \lambda_{\text{backward}} & (\omega_{\text{backward}} - \omega_{\text{dir}}) \end{cases}
$$

$$
\omega_{\text{h}} = \begin{cases} n_{\text{m}}^2 \lambda_{\text{m}} & (\omega_{\text{dir}} - \omega_{\text{h}}) \\ + n_{\text{mstop}}^2 \lambda_{\text{mstop}} & (0 - \omega_{\text{h}}) \end{cases}
$$

Again, we have the time scale relations $\lambda_{\text{faster}}, \lambda_{\text{slower}} \ll \lambda_{\text{fip}} \ll \lambda_{\text{left}}, \lambda_{\text{right}}, \lambda_{\text{forward}}, \lambda_{\text{obst},\omega} \ll \lambda_{\text{m}}, \lambda_{\text{mstop}}$. For the $\omega_{\text{abs}}$ dynamics the corresponding holds as said about the $\nu_{\text{abs}}$ dynamics. If the robot is to move backward, the intuitive interpretation of “left” and “right” changes. Figure 2 illustrates this. If the robot is

---

**Figure 2:** Direction Interpretation when moving backward

It is said “Move backward right”, we would expect it to move backwards and simultaneously turn to the left. This behavior is achieved by the $\omega_{\text{int}}$ dynamics which incorporates $\omega_{\text{abs}}$ as a slowly moving attractor and $n_{\text{mbackward}}$. [Diagram of direction interpretation]
as the parameter that controls sign inversion. The value of \( \omega_{\text{int}} \) couples into the \( \omega_{\text{int}} \) dynamics which sets up \( \omega_{\text{int}} \) depending on the direction specification. Notice that “Turn forward” simply means “Do not turn”. The first two attractors each have an activation factor scaled by a factor \( f_{\text{inhibit}} \). This factor usually equals 1 and decreases to 0 when obstacles or objects arise on either side of the robot and the robot is currently moving. This way, explicitly moving to either side is inhibited if the respective direction contains obstacles. The fourth attractor is generally related to obstacle avoidance and impacts the dynamics only if the robot moves; Turning on the spot should not be a problem, even not in the neighborhood of obstacles. Finally, the \( \omega_{\text{int}} \) dynamics generally handles switching on and off rotation. The activations \( n_t \) and \( n_t_{\text{default}} \) are mutually exclusive by design and are needed both to implement the idea that moving includes turning but turning does not include moving.

The pointing dynamics is nearly trivial and merely handles activation of pointing:

\[
\dot{\phi}_n = n_t^2 \lambda_p \left( \phi_{\text{field}} - \phi_n \right) + n_t \lambda_{\text{stop}} \lambda_{\text{stop}} \left( 0 - \phi_n \right) \tag{7}
\]

If pointing is deactivated, this dynamics has an attractor at \( \phi_n = 0 \). The complexity of pointing and the relation to physical objects is contained in \( \phi_{\text{field}} \), which is the position of a neural field peak. This subject is addressed in section 3.5.

### 3.4 Arbitrator Dynamics

In this subsection we will describe an arbitration scheme based on dynamical systems that is featured by high stability, easy configurability and extensibility. The system we present is a derivative of the systems described in (Steinhage, 1998), (Steinhage and Bergner, 1998) and (Menzen and Steinhage, 1999). It consists of \( N \) dynamical state variables \( n_i \) and associated sensorial contexts \( s_i \). All these quantities might be involved in the decision which behavior to activate and which to deactivate. The following requirements on the arbitration system hold:

- As described in the introduction, a sensor context \( s_i = 1 \) votes for activation of behavior \( i \).
- Behaviors, when becoming active, should be able to activate or deactivate other behaviors. E.g., a behavior left should deactivate a behavior right. Such conditions are referred to as dynamical conditions since they only take effect at the time a behavior comes up, but not later when that behavior is still active. This interrelation between behaviors is coded the Activation Matrix \( A \) and gives rise to an Activation Context \( f_{\text{Activation}} \in [0;1] \).
- Behaviors might be mutually exclusive, or might require one or a set of other behaviors to become active. This dependency is coded in the Logic Matrix \( L \) which in turn gives rise to a Logic Context \( f_{\text{Logic}} \in [0;1] \). These conditions have highest priority.

The arbitration system is based on the following dynamics:

\[
\tau_i \dot{n}_i = \alpha_i n_i - |\alpha_i| n_i^2 + \xi_i \tag{8}
\]

If \( \alpha_i = 1 \), (8) has stable fixed points at \( \pm 1 \). For \( \alpha_i = -1 \), there is one stable fixed point at 0. \( \xi_i \) is an additive noise term of low amplitude which ensures that the system does not by chance get stuck in an unstable fixed point. Systems of type (8) are widely discussed in the literature, see e.g., (Perko, 1991). Next, \( \alpha_i \) is to be computed appropriately as a function of the sensor context \( s \), the system’s state \( n \), the Logic Matrix \( L \) and the Activation Matrix \( A \). Making \( \alpha_i \) a direct instantaneous function of \( n \) is not possible however, since this could lead to instabilities because of the violation of the time scale separation condition. Hence, we first introduce an auxiliary dynamics for \( \alpha \) which tracks a value \( \tilde{\alpha} \) on a slower time scale:

\[
\dot{\tilde{\alpha}}_i = \frac{\alpha_i - \alpha_i}{\tau_\alpha}, \text{ where } \tau_\alpha > \tau_n. \tag{9}
\]

That way \( \alpha_i \) may change instantaneously depending of \( n_i \) while yet preserving the stability of (8). If we define

\[
\tilde{\alpha}_i = 2 f_{\text{Logic},i} f_{\text{Activation},i} - 1, \tag{10}
\]

and consider the limit cases \( f_{\text{Logic},i} = 1 \) and 0 then \( \tilde{\alpha}_i = 1 \) if both \( f_{\text{Logic},i} = 1 \) and \( \tilde{\alpha}_i = -1 \) otherwise.

Every elementary behavior is provided with three indicators that specify their logical relation to the other behaviors. For behavior \( i \) we have

\[
f_{\text{Logic},i} = f_{\text{AND},i} f_{\text{OR},i} f_{\text{NOT},i} \tag{11}
\]

As soon as one of the elementary logical conditions is violated, the whole product is 0.

For \( f_{\text{AND},i} = 1 \), we require that all of the AND-flagged behaviors \( j \) are active. \( f_{\text{OR},i} = 1 \) requires at least one of the OR-flagged behaviors \( j \) to be active. Consequently, \( f_{\text{NOT},i} = 1 \), if none of the NOT-flagged behaviors \( j \) is active. If we define the AND flag as \( L_{ij} = 1 \), the OR-flag as \( L_{ij} = 2 \) and the NOT-flag as \( L_{ij} = 3 \), suitable definitions for the factors are

\[
f_{\text{AND},i} = \prod_{j=1}^{N} \left( 1 - \frac{L_{ij}(L_{ij} - 2)(L_{ij} - 3)}{2} \right), \tag{12}
\]

\[
f_{\text{OR},i} = 1 - \prod_{j=1}^{N} \left( 1 + \frac{L_{ij}(L_{ij} - 1)(L_{ij} - 3)}{2} \right) + \prod_{j=1}^{N} \left( 1 + \frac{L_{ij}(L_{ij} - 1)(L_{ij} - 3)}{2} \right) \tag{13}
\]
\[ f_{\text{NOT},i} = \prod_{j=1}^{N} \left( 1 - \frac{L_{ij}(L_{ij} - 1)(L_{ij} - 2)}{6} n_{ij}^2 \right) \]  

(14)

The functionality of (12), (13) and (14) may seem to be somewhat encrypted but it can be envisioned by choosing a simple setup with \( N = 3 \) and manually computing. As soon as the respective condition is violated, the corresponding factor is 0.

The factor \( f_{\text{Activation},i} \) from Eq. 10 represents activation requests. It is defined as

\[ f_{\text{Activation},i} = 1 - (1 - \rho_i)(1 - \beta_i) \]  

(15)

Obviously, \( f_{\text{Activation},i} = 1 \) as soon as either \( \rho_i = 1 \) or \( \beta_i = 1 \). Here, \( \rho_i \) is a filtered version of the Sensor Context \( s_i \):

\[ \rho_i = \frac{(1 - \rho_i)}{\tau_{\rho,up}} s_i + \frac{0 - \rho_i}{\tau_{\rho,down} - \tau_{\rho,up}} (s_i - 1) \]  

(16)

The \( \rho \) dynamics serves for preprocessing physical sensors and aims at neurological principles like refraction. The signal \( \beta_i \) is related to the activation of behaviors by other behaviors. The first step in implementing the activation / deactivation mechanism through other behaviors is to generate a signal that tracks the ascending flanks of the activation signals \( n_i \). This is achieved by computing a difference signal \( n_{\text{diff},i} = |n_i| - n_{\text{bi}} \), where

\[ n_{\text{bi}} = \frac{1 - n_{\text{bi}}}{\tau_{\rho,up}} |n_i| + \frac{0 - n_{\text{bi}}}{\tau_{\rho,down} - \tau_{\rho,up}} (1 - |n_i|) \]  

(17)

If we choose \( \tau_{\rho,down} = \tau_{\rho} \) and \( \tau_{\rho,up} > \tau_{\rho} \), \( n_{\text{bi}} \) will be somewhat delayed in the rising phase of \( n_i \), but will immediately follow \( n_i \) in the falling phase. Hence, the difference signal will have peaks at the rising flanks of \( n_i \). These peaks are used to push the following bistable dynamics into the desired attractors:

\[ \tau_{\beta_i}' = \frac{2|\beta_i - 1 - (2\beta_i - 1)^3|}{1 + |a_i \cdot (d_i - 1) \cdot (2\beta_i - 1) - d_i \beta_i} \]  

(18)

where \( a_i \) and \( d_i \) are the “(de)activating” forces that depend on \( n_{\text{diff},i} \). This \( \beta \) dynamics is discussed in detail in (Menzner and Steinlage, 1999). For \( d_i = 1 \) the attractor is at \( \beta_i = 0 \) while if \( a_i = 1 \) and \( d_i = 0 \) the attractor is at 1. Obviously the deactivation force has higher priority, \( a_i \) and \( d_i \) are computed according to the following equations:

\[ a_i = f_{\text{Logic},i} \left( 1 - \prod_{j=1}^{N} \left( 1 - \frac{A_{ij}(A_{ij} + 1)}{2} n_{\text{diff},j} \right) \right) \]  

(19)

\[ d_i = f_{\text{Logic},i} \left( 1 - \prod_{j=1}^{N} \left( 1 - \frac{A_{ij}(A_{ij} - 1)}{2} n_{\text{diff},j} \right) \right) \]  

(20)

where \( A_{ij} \in \{-1, 0, 1\} \) are the elements of the Activation Matrix and “1” leads to activation and “-1” to deactivation. Multiplying with \( f_{\text{Logic},i} \) is necessary in order to prevent the \( \beta \) dynamics from “remembering” an activation request of a behavior currently blocked by the logical conditions.

So far, we have completed the description of an arbitration system that is completely configurable by means of two matrices \( \mathbf{L} \) and \( \mathbf{A} \).

### 3.5 Neural Fields for Pointing

In this subsection we will address the question of how to compute the quantity \( \phi_{\text{field}} \) that appears as a linear attractor in Eq. 7. The task of pointing to objects in a scene under influence of keyword speech already is a complex task. We require the system to automatically attend to possibly existing objects, commands like left should be interpreted by the system as to point to the next object on the left hand side and yet the specification of forward should make the robot exactly pointing ahead. If explicitly desired, the robot must be able to point to arbitrary directions even if no objects are located at the respective angles. As the visual sensor presents time varying distributed information, properties like spatial averaging and stabilization in time are required. In conclusion, it turns out that “pointing to objects”, as we understand it, involves some cognitive abilities. Neural fields, if designed appropriately, are able to provide that amount of cognitive capacity. The key concept in our work is that we control peaks of a neural field by means of sensory stimuli. In the following we will show how it is possible to make peaks in a neural field move by introducing an appropriate asymmetry into the kernel and how to control that movement within our arbitration framework depending on sensory input and vision sensor data.

Our considerations are based on the neural field equation as formulated by AMARI (see (Amari, 1977)) with even kernel:

\[ \tau \ddot{u}(x,t) = -u(x,t) + \int_{-\infty}^{\infty} w(x,y) f(u(y,t)) \, dy \]  

\[ + s(x,t) + h \]  

(21)

Furthermore, we assume that (a) the field has relaxed into a so-called a-solution with one peak, (b) the input signal \( s(x,t) = 0 \) and (c) at \( t = 0 \) the kernel is made asymmetric by overlaying an at first arbitrary function, \( w_a = w_v + w_o \). (21) is then read as

\[ \tau \ddot{u}(x,t) = -u(x,t) + \int_{-\infty}^{\infty} w_a(x,y) f(u(y,t)) \, dy + h \]  

(22)

Now, let \( U(x) = u(x,0) \) be the initial stable peak solution. The excitation distribution for any instant later in
time is then

\[ u(x,t) = U \left( x + \int_0^t v(\eta) \, d\eta \right) , \]

where \( v(t) \) is the velocity of the moving peak. Substituting (23) in the left hand side of Eq. 22 yields

\[ \tau \dot{u}(x,t) = \tau U' \frac{d}{dt} \left( x + \int_0^t v(\eta) \, d\eta \right) = \tau U' v(t) \]

where the chain rule has been used to compute the derivative. For the right hand side of (22) we have

\[
- U + \int_{-\infty}^{\infty} w_0(x,y) f(U(y)) \, dy + h \\
= \int_{-\infty}^{\infty} w_0(x,y) f(U(y)) \, dy
\]

where we exploited that for equilibrium solutions \( \int_{-\infty}^{\infty} w_0(x,y) f(U(y)) \, dy = U - h \). Finally, using (24) and (25), Eq. 22 can be rewritten as

\[ \tau U' v(t) = \int_{-\infty}^{\infty} w_0(x,y) f(U(y)) \, dy \]

It becomes clear that the relation between \( w_0 \) and \( v(t) \) is quite complex, especially since it depends on the initial excitation distribution \( U(x) \) and the nonlinearity \( f(\cdot) \). The only solution to overcome this problem is to set \( w_0 = p(t) w_0' \), i.e., a possibly time dependent factor multiplied by the spatial derivative of the symmetric kernel part.

We are then able to exchange the order of derivation and integration and Eq. 26 reads as

\[
\tau U' v(t) = p(t) \frac{d}{dx} \left( \int_{-\infty}^{\infty} w(x,y) f(U(y)) \, dy \right) = p(t) U' \\
= U - h \quad \text{for equilibrium solutions}
\]

So we finally obtain the very simple relation

\[ v(t) = \frac{U(t)}{\tau} \]

Our strategy for the application is now, to introduce asymmetry into the kernel depending on speech input in order to make the peak of the field move from one object, defined by a peak in the proximity encoding input sensor signal, to the next. The kernel is computed according to the following equation:

\[ w_a = w_0 + (|n_{\text{right}}| - |n_{\text{left}}|) c_{\text{flee}} + s C_{\text{es}} c_{\text{relax}} + n_{\text{p, forward}} d C_{\text{straight}} (1 - |n_{\text{ok}}|) w_0 \]

Primarily, the factor \( p(t) \) consists of three additive parts. The first is effective for the very short time when \( n_{\text{right}} = 1 \) or \( n_{\text{left}} = 1 \), that is, when a right or left has just been encountered by the robot. This is called the flee phase because the field “flee’s” from the current object. The arbitrator dynamics ensures that \( n_{\text{right}} = 1 \) and \( n_{\text{left}} = 1 \) cannot happen simultaneously and that at the same time \( n_{\text{p, forward}} = 0 \). The velocity in this phase is parameterized by \( c_{\text{flee}} \). The second phase, called “relax” phase, occurs during movement to the next object. \( C_{\text{es}} \) basically is a threshold value generated from the scalar product of the excitation \( u(x,t) \) and the input \( s(x,t) \). We have \( C_{\text{es}} = 0 \) if the excitaton peak sufficiently overlaps with input and \( C_{\text{es}} = 1 \) otherwise. This means, the peak movement is stopped as soon as the field peak overlaps with the next object, \( s = \pm 1 \) is a factor that codes the direction of the movement in this phase. \( c_{\text{relax}} \) is a constant value and determines the velocity. The third summand serves for setting up fixed pointing directions like ahead (\( \phi_a = 0 \)). The term \( (1 - |n_{\text{ok}}|) \) allows terminating a pointing movement instantaneously by uttering an ok. If the field is designed properly, the peak will remain at that position.

Next, we show an example of what has been described above. In this experiment the robot was located in the scene exactly as shown in Figure 1 and it was commanded with the keyword sequence point - left - right - right - ok. As a consequence the robot started pointing to the next object to the left (which is near its heading direction), to the next object to the right (which is the rightmost object) and again to the right where no more objects are located. A short time later the pointing movement was stopped by the ok. In Figure 3, we show a relevant portion of the time course of the neural field during this process. At the bottom we find the input

![Figure 3: Time course of the neural field during pointing](image-url)
is represented by the gray-shaded surface. It can be seen that the field also shows the input distribution characteristics. This becomes clear when looking at Eq. 21: For equilibrium solutions \( \dot{u}(x, t) = 0 \), \( u(x, t) \) consists of the convolution integral overlayed by the input signal. Furthermore, the figure shows the plane \( u(x, t) = 0 \) by a mesh, since this plane is helpful for the decision of what parts of the field are positive. The time course starts shortly before the first right is uttered. After the right has been received, the field moves its peak to the objects next to the right. At the time of the next right, the field peak again starts to move to the right, but this time there is no object to “snap in” so that the movement keeps on. A final ok terminates the movement by raising \( n_{ok} = 1 \) and thus setting \( p(t) = 0 \). It appears that the peak distant of objects is not as high as in objects; this is again caused by the overlayed input signal. But it should be noted that also at places where no input resides, the peak solution remains stable so that at all times we have a well defined \( \phi_{\text{field}} \) in Eq. 7. One more important property of the neural field is illustrated in Figure 4. This experiment was executed under the same circumstances. Additionally, we overlayed a tumbling rotation on the platform so that, effectively, the objects seemed to move slowly. Figure 4 shows that even under these circumstances the field tracks the objects and is able to switch between the objects. This example shows that the approach provides much potential with respect to sensor coupled grasping and speech interpretation. The dropouts in the peak course at the transitions between the objects really are artefacts of subsampling in time for the sake of readability. In fact, the peak never falls short of zero.

3.6 Vision Sensor Processing

In this section we will explain how the vision sensor is evaluated so that the results fit into the main behavioral dynamics as well as into the neural field. The vision sensor is a depth profile of the frontal semi circle, sampled at 73 positions (see the bottom area of Fig. 1). This results in a vector of distance samples \( \mathbf{d} = \{d_i\} \) where \(-31 \leq i \leq 31\). For the neural field the reciprocal values \( 1/d_i \) are evaluated and a threshold operation is applied so that there exist maxima where objects are located. This yields sensor distributions as for example shown in Figures 3 and 4. For the main behavioral dynamics (see section 3.3) it is still left to describe how \( \lambda_{\text{obst,v}}, \lambda_{\text{obst,ω}}, v_{\text{obst}}, ω_{\text{obst}}, f_{\text{finsh, left}} \) and \( f_{\text{finsh, right}} \) are computed. Most trivially, \( \lambda_{\text{obst,v}} \) and \( \lambda_{\text{obst,ω}} \) become positive if

\[
d_{\text{min}} = \min\{i | d_i \cos \phi_i < |\gamma_{\text{Robot}}|\}
\]  

(29)

falls below a certain limit value. (29) concretely means, we examine that segment of the obstacle distribution that would obstruct movement into the current heading direction and inspect the minimum of these values. Next, a “gravity center” and an average value are extracted according to

\[
g_c = \frac{\sum_{i=-R_0}^{R_0} i d_i}{\sum_{i=-R_0}^{R_0} d_i}
\]  

(30)

\[
g_a = \frac{1}{2R_0 + 1} \sum_{i=-R_0}^{R_0} \frac{1}{d_i}
\]  

(31)

The angular velocity attractor is then computed as

\[
\omega_{\text{obst}} = \frac{V_{\text{max}}}{g_c} \left( \frac{\omega_{\text{dist}}}{\sqrt{g_c^2 \theta_{\text{min}}}} \right)
\]  

(32)

There is one factor that takes the current velocity into account, one that merges direction information and an exponential term that introduces damping due to obstacle laterality and distance. Damping is parameterized by the two constants \( D_{\text{dist}} \) and \( D_{\text{dist}} \). Appropriate computation of \( v_{\text{obst}} \) is achieved using a Time-To-Contact (TTC) approach. Given the current velocity, \( v_h \), and the distance according to Eq. 20, we assume that we decelerate with constant negative acceleration \( a_{\text{TTC}} \) so that the velocity vanishes at the time the collision would occur. With

\[
a_{\text{TTC}} = -\frac{v_h^2}{2d_{\text{min}}}
\]  

(33)

we obtain

\[
v_{\text{obst}} = a_{\text{TTC}} t_{\text{Euler}}
\]  

(34)

Figure 4: The neural field tracks moving objects.
Here, $t_{\text{Euler}}$ is the Euler step width of our system.

For the two inhibition factors $f_{\text{inhib\_left}}$ and $f_{\text{inhib\_right}}$ we have the constraints that they must equal 1 for pure turning, and that they must vanish if the robot moves forward and obstacles arise on the respective side. To implement a nonlinear onset of obstacle contribution at a certain intensity we use a sigmoid function:

$$f_{\text{inhib\_left}} = \eta^2 + \eta_{\text{mov\_forward}}^2 \left( \frac{1}{2} - \frac{1}{2} \tanh(-\eta_{\text{mov\_left}} + T_0) \right)$$

(35)

where $\eta_{\text{mov\_left}}$ is the integrated obstacle intensity on the respective side and $T_0$ is the threshold from which on inhibition occurs.

### 3.7 Verification and Results

We now describe a representative sample run started in the scene shown in Figure 1. The annotated path is shown in Figure 5. An overview how the activities $n_{...}$ being cyclic, is plotted so that if it exceeds the lower bound it is continued at the upper bound and vice versa.

JAVA\textsc{robot} initially gazed at the five small objects, when it was told “(1) move - (2) forward”. While the activity of $n_{\text{forward}}$ vanishes shortly after, $n_{\text{mov\_default}}$ stays active because this activity is not only related to a keyword, but also indicates that the robot is in movement mode. $n_{\text{mov\_default}}$ keeps itself active by means of the activation mechanism described in section 3.4. It is also important to note that $n_{\text{mov\_default}}$ has been activated so that the robot is not restricted to straight ahead movement. The robot starts to move ahead and although not explicitly specified, a turning component arises due to the obstacles. This can be seen by watching the robot’s path, but also by noting that $\phi_h$ changes to the right at $t = 25...35$. The next two keywords “(3) left - (4) forward” make the robot approach the wall. At $t = 75$, the wall is near enough to take effect and the heading direction is adjusted to about $\phi_h = 0^\circ$ via $\omega_{\text{obst}}$. At $t \approx 100$ the corner is near enough to let the robot again update its heading direction. No verbal intervention is necessary for all this. To reflect the complete path with respect to activities and obstacle contributions is left as an exercise for the patient reader. There is, however, one more interesting point that shows up at $t \approx 210$ where the robot is commanded “(7) right”. At this point it is not yet possible to move right because that direction is obstructed by the object. Now, inhibition via $f_{\text{inhib\_right}}$ comes into play and defers moving to the right until the obstacle is passed. This is also visible in Figure 6 where the current $\phi_h$ is retained until $t \approx 225$, although $n_{\text{mov\_left}}$ has already been active for some time.

The sample run discussed so far is representative for the navigation part of the system, which implements sensor fusion of keyword speech and (visual) depth informa-

![Figure 6: Activities and Heading direction over time for the sample run](image-url)
tion. At any time the system can be switched to pointing. This is achieved by the keyword point. In this mode the system autonomously determines the pointing direction depending on the currently fixated objects and the keywords left and right are automatically interpreted as commands to point to the next object into the respective direction. In addition, the keyword ok may be used to let the system keep the current pointing direction, no matter whether it currently points to an object or not. Control of the pointing angle is implemented using dynamic neural fields as described in Section 3.5. The behavior of the field under the constraints of the object distribution and the keyword has already been visualized in Figure 3 for a sample experiment in the scene given by Figure 1. Each transition from one object to the next involves of transition from a symmetric to an asymmetric kernel and back. Figure 3 also has illustrated that the peak remains stable at angles where no input resides, that is, where no objects are located. A further important feature of the system has been shown in Figure 4: The system is able to track slowly moving objects while still being able to switch between these objects. Put another way, the system really brings about object oriented behavior instead of behavior oriented at a mathematical quantity like an angle.

4. Conclusions

We have presented a new approach that is applicable to various problems that arise in the field of behavior generation in the context of autonomous robotics. The capacity of the approach has been shown in an example. The approach introduces a "toolbox" of solution techniques, all based on dynamical systems. As opposed to using hybrid systems, we can guarantee the stability of our system by obeying easy to state time scale separation conditions. By nature, continuous dynamical systems are well suited for linking, as a control system, between sensors and effectors since both sensors and effectors may also be considered as dynamical systems. The concept of time scale covers switching (section 3.4) and smooth transitions of dynamic quantities (section 3.3).

In this paper we have presented the application of the system in a simulation environment. However, large parts of it have already been used to control movement and pointing direction of a real robot. A detailed description of this work can be found in (Menzner and Steinhaeuser, 1999) where the robot ARNOLD is described and a photo series guiding ARNOLD via speech commands is presented. This paper also shows that our design principle—to consider the actual robot as an exchangeable device expecting a small set of control variables—proves well in practice. See also (Steinhaeuser and Bergener, 1998) for another robot implementation of the dynamic framework for behavioral organization.

Extensions towards learning and internal simulation of behavior sequences are already on the way. A practical application is given in (Steinhaeuser and Bergener, 2000).

Finally we would like to emphasize that the current paper is not intended to be the (n+1)st publication about navigation and control of autonomous robots. Rather, our work aims at developing a unified mathematically clean framework based on dynamical system with applicability to all problems that occur in context with robotics and sensor processing, actuator control, behavior generation and learning.

References


