Temporal Coordination among Two Vision-Guided Vehicles: A Nonlinear Dynamical Systems Approach

Cristina P Santos and Manuel João Ferreira
Industrial Electronics Department, University of Minho, Guimarães Portugal

1. Introduction

The field of multiple autonomous robots cooperating is emerging as a key technology in mobile robots and is currently under intense effort. The use of multi-robots synchronized, coordinated or cooperating in production processes where there is a high requirement on flexibility and manoeuvrability is highly desirable. This is an option to be considered in complex and integrated production processes including assembling, transporting, painting and welding tasks.
Broadly, the applied general approaches for controlling and coordinating the movement of several robots that cooperatively perform a task illustrate the major trade-off in the control and coordination of multi-robots: between precision and feasibility and between the necessity of global information and communication capacity. Further, multiple robot systems working in external synchronization, e.g master-slave and coordinated schemes or mutual synchronization, e.g. cooperative schemes, imply the design of suitable controllers to achieve the required synchronous motion.
The work presented in this paper, combines insights of computer vision, dynamical systems theory, computational neuroscience and robotics. We aim at generating online flexible timed behavior stably adapted to changing online visual, infrared and proprioceptive sensory information, such that different entities may achieve autonomous timed and flexible cooperative/coordinated behavior. As a first attempt, we do not take into account communication issues. We apply an attractor based dynamics as recent studies have shown that this theory helps synchronize systems and reduces the computational requirements for determining identical movement parameters across different coupled entities. The inherent advantages from an engineering viewpoint are huge, since the control system is released from the task of recalculating the movement parameters of the different entities.
The main motivation is that once solutions for this problem are found, they can be applied in search and rescue operations, landing removal, remote terrain and space exploration, and also to the control of satellites and unmanned aerial vehicles. In this domain, the achievement of robots able to exhibit intelligent and flexible cooperative behaviour is a first issue.
The approach is demonstrated in the cooperation among two vision-guided mobile robots such that they are able to reach a visually acquired goal, while avoiding obstacles, without

Source: Computer Vision, Book edited by: Xiong Zhihui,
ISBN 978-953-7619-21-3, pp. 538, November 2008, I-Tech, Vienna, Austria

prior knowledge of the non-structured and complex environment. Both systems have to deal with time constraints, such that vehicles have to reach the goal location within a certain time independently of the environment configuration or the distance to the target. Goal position is acquired by a camera mounted on the top of the robot and facing in the direction of the driving speed.
The results illustrate the robustness of the proposed decision-making mechanism and show that the two vehicles are temporal coordinated: if a robot movement is affected by the environment configuration such that it will take longer to reach the target, the control level coordinates the two robots such that they terminate approximately simultaneously.
To the best of our knowledge, temporal coordination among robots able to deal with both space and time constraints, has not been addressed in the framework of dynamical systems except from our previous work. The system is novel because coordination (to synchronize or to sequentialize) autonomously results from the current sensorial context through an adaptive process that is embedded in the dynamical systems controller. Online modification of the parameters is used to steer action and feedback loops enable to do online trajectory modulation. We also attempt to demonstrate that the approach can be extended for a larger number of vehicles.
In the rest of the article, we will first give a brief review of the state of the art of trajectory planning considering time control. A brief discussion of the proposed method and its advantages is done in this section. In section 3 we present the dynamical systems approach used to generate timed trajectories, formulate the proposed controller and discuss its intrinsic properties. In the next section, we describe the problem we try to solve in this chapter. The overall architecture is presented. We also describe the dynamical systems that act at the level of heading direction, the vision system, the behavioural specifications and control of forward velocity. and the other controlling the robot's velocities. In this section, it is also described the vision system. We then describe the simulation environment for the application, we present two simulations and our results and discuss the properties of the system. We conclude by presenting the conclusions and presenting future directions for the work (section 6).

2. State-of-the-art

Trajectory planning has been extensively studied over the last few years, ranging from the addition of the time dimension to the robot's configuration space (Erdmann & Lozano-Perez, 1987), visibility graph (Reif & Sharir, 1985), cell decomposition (Fujimura & Samet, 1989) or neural networks (Glasius et al., 1994). There are several results for time-optimal trajectory planning (Fraichard, 1999).
Despite the efficient planning algorithms that have been developed and the advances in the control domain which validated dynamic, robust and adaptive control techniques, the path planning problem in autonomous robotics remains separated in theory from perception and control. This separation implies that space and time constraints on robot motion must be known before hand with the high degree of precision typically required for non-autonomous robot operation. In order to develop autonomous robot systems capable of operating in changing and uncertain environments it is required a tight coupling of planning, sensing and execution.
However, timing is more difficult to control when it must be compatible with the requirement of continuous coupling to sensory information. Some approaches have addressed this issue (Buhler & Kindlmann, 1994), but timing was not fully explored.

In this article, we propose an approach fully formulated in terms of nonlinear dynamical systems which lead to a flexible timed behaviour stably adapted to changing online sensory information. Dynamical systems have various desirable properties which makes them interesting and powerful for trajectory generation. See (Schoner & Dose, 1992; Tani et al., 2004; Schaal et al., 2001; Schoner & Santos, 2001; Fukuoka et al., 2003; Ijspeert et al., 2001) for related work. First, the structural robustness of the solutions implies intrinsic robustness against small perturbations and noise and the possibility to fuse new inputs into the system without completely destroying its properties. Second, the low computation cost is well-suited for real time. Other properties are the smooth online modulation of the trajectories through changes in the parameters of the dynamical systems; the possibility to synchronize with external signals and to add sensory feedback pathways. The dynamics of the system globally encode a task (i.e. the whole attractor landscape) with the goal state as the point attractor. This is a "always online" property, i.e., once a task is encoded into a dynamical system (e.g. learning) it will be always active, and no discrete trials are needed. Once properly designed, the dynamical system can be robust enough against perturbations and able to smoothly recover from perturbations by means of coupling terms in the dynamics. Another particularity is that these systems produce coordinated multidimensional rhythms of motor activity, under the control of simple input signals. Such systems are deemed to strongly reduce the dimensionality of the control problem.

We build on previous work (Santos, 2004; Schoner & Santos, 2001; Schoner, 1994), where we proposed a dynamical system architecture that generated timed trajectories, including rhythmic and discrete movement, movement sequences and temporally coordinated movements. The model consists of a dynamical system composed of stable fixed points and a stable limit cycle (an Hopf oscillator). Trajectories are generated through the sequencing of these primitives, in which the limit cycle is activated over limited time intervals. This sequencing is controlled by a “neural" competitive dynamics. By controlling the timing of a limit cycle, the system performs well tasks with complex timing constraints. The online linkage to noisy sensorial information, was achieved through the coupling of these dynamical systems to time-varying sensory information (Schoner, 1994; Santos, 2004). In (Santos, 2004), this architecture was implemented in a real vehicle and integrated with other dynamical architectures which do not explicitly parameterize timing requirements. In (Schoner & Santos, 2001), we have generated temporally coordinated movements among two PUMA arms by coupling two such dynamical systems.

In this work, coordination is modeled through mutual coupling of such dynamical systems. This coupling enables to achieve temporal coordination and synchronization of the different systems, providing an independency relatively to the specification of their individual parameters. Specifically, we address the following questions: Can the temporal coordination among different degrees-of-freedom (dofs) be applied to the robotics domain such that a tendency to synchronize among two vehicles is achieved? Can the applied dynamical systems approach provide a theoretically based way of tuning the movement parameters such that it is possible to account for relationships among these?

These questions are positively answered and shown in exemplary simulations in which two low-level vehicles must navigate in a simulated non-structured environment while being capable of reaching a target in an approximately constant time. For each robot, target position is internally acquired by a visual system mounted over the robot and robot velocity is controlled such that the vehicle has a fixed time to reach the target while continuously avoiding sensed obstacles in its path. The two robot movements are coupled in time such that if the two movements onsets are not perfectly simultaneous or if their time trajectories are evolving differently (one is going faster/slower than the other), leading to different movement times (time it takes to reach the target), this coupling coordinates the two movements such that they terminate approximately simultaneously.

Interesting properties of the system include: 1) the possibility to include feedback loops in order to do online trajectory modulation and take external perturbations into account, such that the environmental changes adjust the dynamics of trajectory generation; 2) online modulation of the trajectories with respect to the amplitude, frequency and the midpoint of the rhythmic patterns (discrete movements goal), while keeping the general features of the original movements, and 3) the coordination and synchronization among the robots, achieved through the coupling among the dynamics of each robot, that provides for a smooth and an adaptive behaviour of the complete system in face perturbations in the sensed environment. This type of control scheme has a wide range of applications in multi-dimensional control problems.

It is our belief that planning in terms of autonomous nonlinear attractor landscapes promises more general movement behaviours than traditional approaches using time-indexed trajectory planning. Further, by removing the explicit time dependency one can avoid complicated 'clocking' and 'reset clock' mechanisms.

3. The dynamical systems trajectory generator

Our aim is to propose a controller architecture that is able to generate temporally coordinated trajectories for two wheeled vehicles such that they reach in time a visually acquired target, independently of the environment configuration or the distance to the target. These trajectories should be smoothly modulated both individually and in coordination when simple control parameters change.

We build on a previously proposed solution in which timed trajectories were generated as attractor solutions of dynamical systems (Santos, 2004). The controller is modelled by a dynamical system that can generate trajectories that have both discrete and rhythmic components. The system starts at an initial time in an initial discrete position, and moves to a new final discrete position, within a desired movement time, and keeping that time stable under variable conditions. The final discrete position and movement initiation change and depend on the visually detected target, on the environment configuration and its perception, on proprioceptive data and on the robot internal model. Thus, trajectories generated by this architecture are modulated by sensory feedback.

The overall controller architecture is depicted in Fig. 1.

In this section we describe the dynamical system architecture that generates timed trajectories. First, we describe the dynamical systems composed of stable fixed points and a stable limit cycle (an Hopf oscillator). The solutions of these dynamical systems are temporally coordinated through the coupling of these architectures. Second, the “neural” dynamics that control the sequencial activation of these dynamic primitives is described. Finally, we discuss some relevant properties of the overall system that enables to achieve generation and temporal coordination of complex movements.

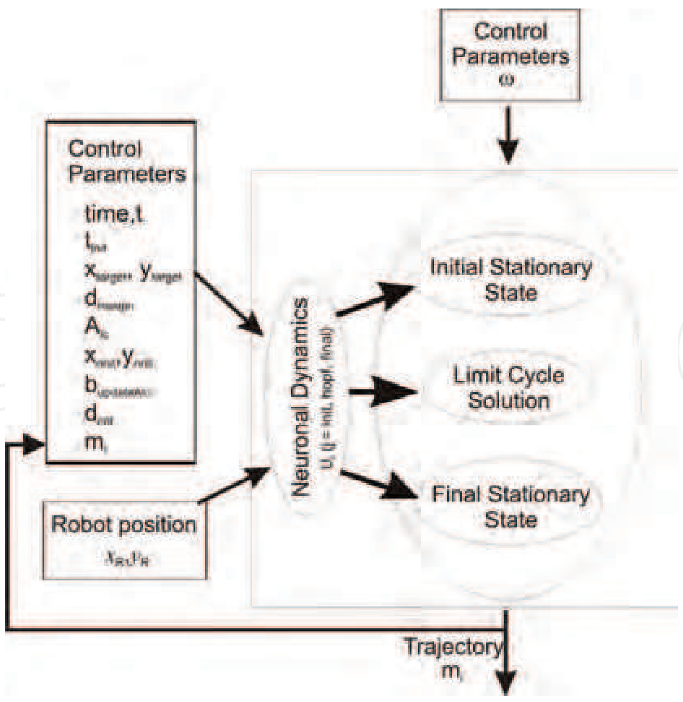

Fig. 1. Controller architecture for timed trajectory generator. Timed movement for *x* and *y* spatial coordinates are generated through the sequencing of stable fixed points and a stable limit cycle. This sequencing is controlled by a neural competitive dynamics according to sensorial context and logical conditions. Trajectories are modulated according to the A_{ic} parameter.

3.1 Fixed points an limit cycle solutions generator

The developed controller is divided in three subsystems, one generating the initial discrete part of movement, another generating the oscillatory part and another generating the final discrete part of movement. A dynamical system for a pair of behavioral variables *(m,n)* is defined to generate the timed movement (Santos, 2004; Schoner & Santos, 2001). Although only the variable, *m*, will be used to set the robotic variable, a second auxiliary variable, *n*, is needed to enable the system to undergo periodic motion.

This dynamical system can operate in three dynamic regimes that correspond to the stable solutions of the individual dynamical systems: two discrete states (stationary states) and a stable oscillation (a limit cycle solution). We set two spatially fixed coordinates systems each centered on the initial robot position: one for the *x* and the other for the *y* spatial coordinates of robot movement. A dynamical system is defined for each of these fixed coordinate systems as follows:

$$\begin{pmatrix} \dot{m}_i \\ \dot{n}_i \end{pmatrix} = 5|u_{init,i}|\begin{pmatrix} m_i \\ n_i \end{pmatrix} + |u_{hopf,i}| f_{hopf,i} + 5|u_{final,i}|\begin{pmatrix} m_i - A_{ic} \\ n_i \end{pmatrix} + gwn, \tag{1}$$

where the index $i = x, y$ refers to *x* and *y* spatial fixed coordinate systems of robot movement.

The "init" and "final" contributions describe a discrete motion whose solutions converge asymptotically to a globally attractive point at $m_i = 0$ for "init" and A_{ic} for "final" with $n_i = 0$ for both. Speed of convergence is controlled by σ = 1/5 = 0.2. time units. If only the final

contribution is active ($u_{hopf} = u_{init} = 0$; $|u_{final}| = 1$), each time A_{ic} is changed, the system will be attracted by the new A_{ic} value, generating a discrete movement towards A_{ic}.
The "Hopf" term describes an Hopf oscillator, that generates the limit cycle solution (as defined in (Santos, 2004; Schoner & Santos, 2001) and is given by:

$$f_{hopf,i} = \begin{pmatrix} \alpha & -\omega \\ \omega & \alpha \end{pmatrix} \begin{pmatrix} m_i - \frac{A_{ic}}{2} \\ n_i \end{pmatrix} - \gamma_i \; ((m_i - \frac{A_{ic}}{2})^2 + n_i^{\,2}) \begin{pmatrix} m_i - \frac{A_{ic}}{2} \\ n_i \end{pmatrix} \tag{2}$$

where $\gamma_i = 4\ \alpha\ /\ A_{ic}^2$ controls the amplitude of the oscillations, ω is the oscillator intrinsic frequency and α controls the speed of convergence to the limit cycle. This oscillator in isolation ($u_{init} = u_{final} = 0$; $|u_{hopf}| = 1$) , contains a bifurcation from a fixed point (when α <0) to a structurally stable, harmonic limit cycle with radius $A_{ic} = sqr(\alpha/\gamma_i)$ cycle time $\Gamma = 2\ \pi/\omega = 20$ time units and relaxation to the limit cycle given by $1/(2\ \alpha\ \gamma_i) = 0.2$ time units, for $\alpha > 0$. Thus, it provides a stable periodic solution (limit cycle attractor)

$$m_i(t) = \frac{A_{ic}}{2} + \frac{A_{ic}}{2}\sin(\omega\, t). \tag{3}$$

The fixed point m has an offset given by $A_{ic}\ /\ 2$. For $\alpha < 0$ the system exhibits a stable fixed point at $m = A_{ic}\ /\ 2$.
Because the system is analytically treatable to a large extent, it facilitates the smooth modulation of the generated trajectories according to changes in the frequency, amplitude or offset parameters. This is interesting for trajectory generation in a robot.
Basically, this Hopf oscillator describes a rhythmic motion which amplitude of movement is specified by A_{ic} and its frequency by ω.
The dynamics of (Eq. 1) are augmented by a Gaussian white noise term, *gwn*, that guarantees escape from unstable states and assures robustness to the system.
The system is able to cope with fluctuations in amplitude A_{ic} because quantities that depend on sensory information are included in the vector field. Lets consider the dynamical systems defined for the x spatial coordinate. The periodic motion's amplitude, A_{xc}, is updated during periodic movement each time step as follows,

$$A_{xc} = (x_{target} - x_{Rinit}) - \big((x_R - x_{Rinit}) - m_x\big), \tag{4}$$

where x_{target} is x target position, x_R is x robot position, x_{Rinit} is initial x robot position previously to movement initiation and m_x is the dynamical variable. We always consider movement is relative to the origin of an allocentric reference frame, which is coincident with x_{Rinit}. Online trajectory modulation is achieved through the inclusion of this feedback loop that enables to take robot movement and environment configuration into account, such that when a change occurs, the system online adjusts the dynamics of trajectory generation. The same behavior applies for the dynamical systems defined for the y spatial coordinate.
Here an approach is defined to achieve temporal coordination among the two robots, by coupling these two architectures in a way that generates phase-locking (synchronization) in the oscillation regime. This was achieved by modifying the *Hopf* contribution that generates the limit cycle solution (Eq. 2) as follows:

$$f_{hopf,i} = \ldots + c\,|\,u_{hopf,j}\,| \begin{pmatrix} \cos\theta_{ij} & -\sin\theta_{ij} \\ \sin\theta_{ij} & \cos\theta_{ij} \end{pmatrix} \begin{pmatrix} m_j \\ n_j \end{pmatrix} \tag{5}$$

where index *j* refers to index *i* time courses of the coupled dynamical system (the other robot) and θ_{ij} is the desired relative phase among oscillators *i* and *j* ($-\theta_{ij}$ among oscillators *j* and *i*). For instance, (m_x, n_x) of robot 1 is coupled with (m_x,n_x) of robot 2. The coupling term is multiplied with the neuronal activation of the other system's Hopf state so that coupling is effective only when both components are in the oscillation regime. Because we want both coupled dynamical systems to be in-phase we set $\theta_{ij} = 0$ degrees.
A neural dynamics controls the switching between the 3 possible modes of movement through three "neurons" $u_{j,\ i}$ (*j =init, hopf, final*). This switch is controlled by several parameters including calculated target position, acquired by the vision system. Moreover, the amplitude A_{ic} of movement depends on the calculated target position and this provides for online trajectory modulation. By modifying on the fly these parameters, one can easily generate different stable trajectories.

3.2 Neural dynamics

The "neuronal"' dynamics of $u_{j,i} \in [-1,1]$ (*j = init, final, hopf; i = x,y* refers to *x* and *y* spatial fixed coordinate systems of robot movement) switches the dynamics from the initial and final stationary states into the oscillatory regime and back. Thus, a single discrete movement act is generated by starting out with neuron $|u_{init,i}| = 1$ activated, the other neurons deactivated ($|u_{final,i}| = |u_{hopf,i}| = 0$), so that the system is in the initial stationary state ($m_i=0$). Then, neuron $|u_{init,i}| = 0$ is deactivated and neuron $|u_{hopf,i}| = 1$ activated and the system evolves along the oscillatory solution. After approximately a half-cycle of the oscillation, this oscillatory solution is deactivated again turning on the final postural state instead ($|u_{final,i}| = |u_{hopf,i}| = 1$). Temporally discrete movement is autonomously generated through a sequence of neural switches such that an oscillatory state exists during an appropriate time interval of about a half-cycle. This approximately half-cycle is movement time (*MT*).
These switches are controlled by the following competitive dynamics

$$\alpha_u \dot{u}_{j,i} = \mu_{j,i} u_{j,i} - \left|\mu_{j,i}\right| u_{j,i}^3 - 2.1 \sum_{a,b \neq j} \left(u_{a,i}^2 + u_{b,i}^2\right) u_{j,i} + gwn \tag{6}$$

where "neurons", $u_{j,i}$, can go "on" (=1) or "off" (=0). The first two terms of the equation represent the normal form of a degenerate pitchfork bifurcation: A single attractor at $u_{j,i} = 0$ for negative $\mu_{j,i}$ becomes unstable for positive $\mu_{j,i}$, and two new attractors at $u_{j,i} = 1$ and $u_{j,i} = -1$ form. We use the absolute value of $u_{j,i}$ as a weight factor in (Eq 1).
The third term is a competitive term, which destabilizes any attractors in which more than one neuron is "on". For positive $\mu_{j,i}$ all attractors of this competitive dynamics have one neuron in an "on" state, and the other two neurons in the "off" state (Schoner & Dose, 1992; Large et al., 1999). The dynamics of (Eq. 6) are augmented by the Gaussian white noise term, *gwn*, that guarantees escape from unstable states and assures robustness to the system.
Fig. 2 presents a schematic illustrating this dynamics. This dynamics enforces competition among task constraints depending on the neural *competitive advantages* parameters, $\mu_{j,i}$. As the environmental situation changes, the competitive parameters reflect by design these changes causing bifurcations in the competitive dynamics. The neuron, $u_{j,i}$, with the largest competitive advantage, $\mu_{j,i} > 0$, is likely to win the competition, although for sufficiently small differences between the different $\mu_{j,i}$ values multiple outcomes are possible (the system is multistable) (Large et al., 1999).

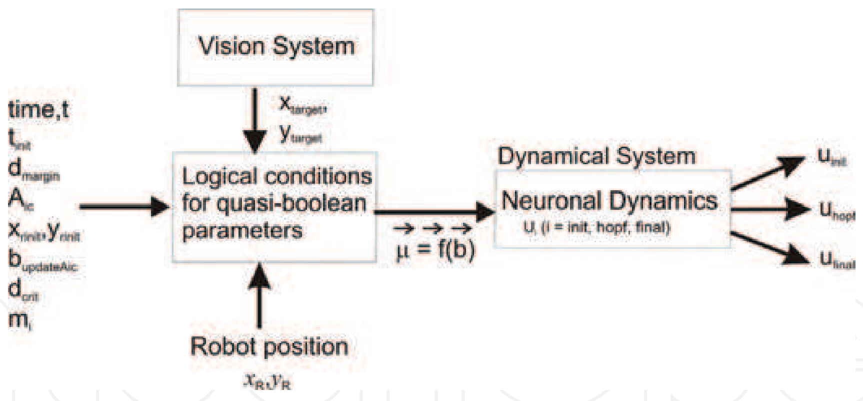

Fig. 2. Schematic representation of the neural dynamics. Current sensorial context and global constraints change as the environmental situation changes. By design, the $\mu_{j,i}$ parameters reflect these changes causing bifurcations in the neural dynamics and activation of a neuron $u_{j,i} \in [-1,1]$. These neurons enable the system to appropriately control sequencing of movement primitives.

In order to control switching, the $\mu_{j,i}$ parameters are explicitly designed such that their functions reflect the current sensorial context and the global constraints expressing which states are more applicable to the current situation. They are defined as functions of robot position, parameters returned by the visual system, information from the other robot and internal states and control the sequential activation of the different neurons (see (Steinhage & Schoner, 1998)), for a general framework for sequence generation based on these ideas and (Schoner & Santos, 2001) for a description). Herein, we vary the μ-parameters between the values 1.5 and 3.5: $\mu_{j,i} = 1.5 + 2\, b_{j,i}$, where $b_{j,i}$ are "quasi-boolean" factors taking on values between 0 and 1 (with a tendency to have values either close to 0 or close to 1). Hence, we assure that one neuron is always "on".

The time scale of the neuronal dynamics is set to a relaxation time of $\sigma_{uj,i} = 1 / \alpha_u = 0.02$, ten times faster than the relaxation time of the (m_i, n_i) dynamical variables. This difference in time scale guarantees that the analysis of the attractor structure of the neural dynamics is unaffected by the dependence of its parameters, $\mu_{j,i}$ on the dynamical variable, m_i, which is a dynamical variable as well. Strictly speaking, the neural and timing dynamics are thus mutually coupled. The difference in time scale makes it possible to treat m_i as a parameter in the neural dynamics (adiabatic variables). Conversely, the neural weights can be assumed to have relaxed to their corresponding fixed points when analyzing the timing dynamics (adiabatic elimination). The adiabatic elimination of fast behavioral variables reduces the complexity of a complicated behavioral system built up by coupling many dynamical systems (Santos, 2005; Steinhage & Schoner, 1998). By using different time scales one can design the several dynamical systems separately.

3.3 Intrinsic properties of the overall dynamics

The fact that timed movement is generated from attractor solutions of nonlinear dynamical systems leads to a number of desirable properties for trajectory generation. The system is able to make decisions such that it flexibly responds to the demands of any given situation while keeping timing stable. Intrinsic stability properties are inherent to the Hopf oscillator, which has a structurally stable limit cycle. Thus, the generated trajectories are robust to the presence of noise and stable to perturbations. This property is specially useful for adding feedback pathways because sensory information is forgotten as soon as it disappears from the environment. This structural robustness of solutions further guarantees the stability and controllability of the overall system if the time scale separation principle is obeyed. These intrinsic properties, including bifurcation and hysteresis, enable planning decisions to be made and carried out in a flexible, yet stable way, even if unreliable sensory information is used to steer action. These properties are explained in more detail in (Santos, 2004).

An advantage of this approach is that it is possible to parameterize the system by analytic approximation, which facilitates the specification of parameters. Not only we have generated discrete movement as well as we provide a theoretically based way of tuning the dynamical parameters to fix a specific movement time or extent. Smooth trajectory online modulation of the trajectories with respect to the goal, amplitude and frequency is now possible, while keeping the general features of the original movements. Trajectories are thus modulated according to the environmental changes, such that action is steered by online modulation of the parameters. A simple modulation of the parameters can generate an infinite variation of stable trajectories.

Moreover, we showed that it was easy to couple two dynamical systems to generate coordinated multidimensional trajectories. The extension to a more enlarged number of dynamical systems is feasible and brings no added complications. The coordination and synchronization among the generated trajectories, achieved through the coupling of their dynamical systems, provides for a smooth and an adaptive behavior of the complete system in face of perturbations in the sensed environment. The coupling of nonlinear oscillators offers multiple interesting properties which enable smooth integration of their parameters and makes them interesting and powerful for trajectory generation.

In the next section, we show the application of this dynamical architecture to the generation of timed trajectories for two vision-guided vehicles.

4. Problem statement

In this article we try to solve a robotic problem applying an attractor based dynamics to timing and coordination. Fig. 3 depicts the problem setup: two low-level vehicles must navigate in a simulated non-structured environment while being capable of reaching a

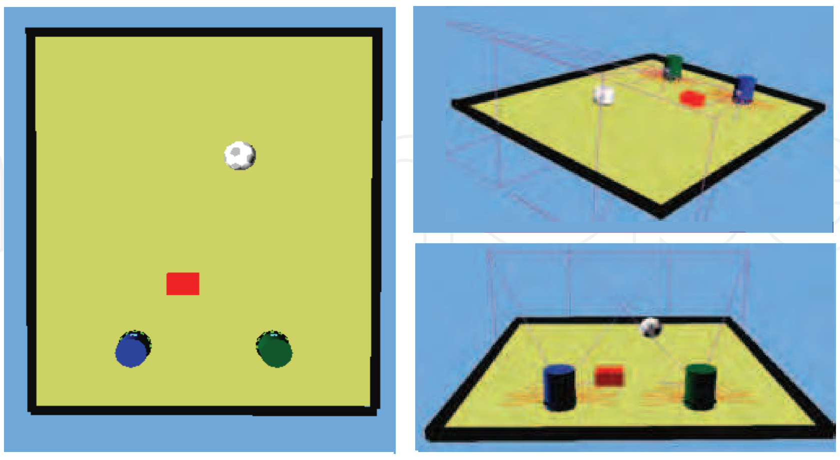

Fig. 3. Three views of a same scenario where two low-level vehicles navigate in a simulated non-structured environment. Each robot moves, senses obstacles and acquires target information through online visual sensory information. Each robot movement is controlled such that target locations are reached in a certain fixed time while avoiding obstacles in its path. The robot controllers are coupled such that their movements are temporally coordinated.

target within a certain time independently of the environment configuration or the distance to the target. Each robot moves, senses obstacles and acquires target information through online visual sensory information. Each robot movement is controlled such that target locations are reached in a certain fixed time while avoiding obstacles in its path. Thus, if the vehicle takes longer to arrive at the target because it needed to circumnavigate an obstacle, this change of timing must be compensated for by accelerating the vehicle along its path.
The task is to temporally coordinate the timed movements of both robots, meaning that if one robot movement is affected by the environment configuration such that it will take longer to reach the target, this robot has to be accelerated and the other robot de-accelerated such that they terminate approximately simultaneously.

4.1 Overall architecture

The overall system architecture is depicted in fig. 4.

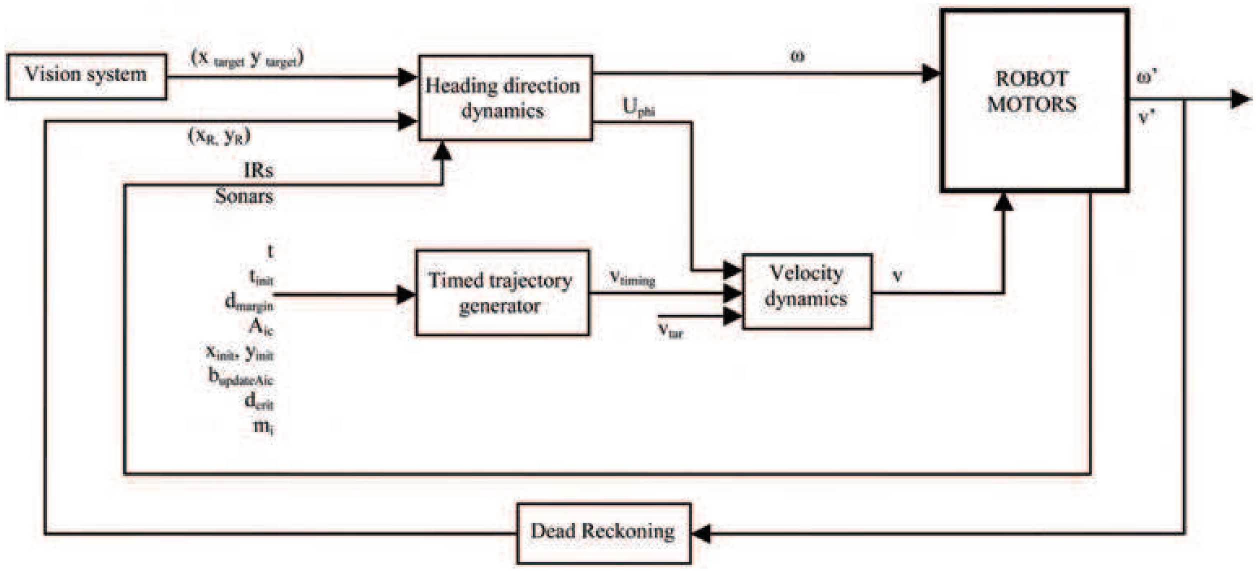

Fig. 4. The overall architecture of the system. Visual acquired target position, robot position and other internal data are transformed onto time-varying parameters of a trajectory controller. An heading direction dynamics acts out at the level of the turning rate and generates angular velocity, *ω*. Forward velocity, *v*, considering timing constraints is generated by a timed trajectory controller. A forward kinematics model translates these velocities into the rotation speeds of both wheels and sent to the velocities servos of the two motors.

At $t = 0$ s, the robot is resting at its initial fixed position, (x_{Rinit}, y_{Rinit}). The robot rotates in the spot in order to orient towards or look for the target direction, which is internally acquired by a visual system mounted over each robot. At time t_{init}, forward movement is initiated.
Forward movement in real time is generated by a controller formulated in terms of nonlinear dynamical systems for dynamical variables $(\phi_h, (m_i, n_i))$. The controller is divided onto two integrated architectures which act out at different levels. The dynamics of heading direction act out at the level of the turning rate (ϕ_h). The dynamics of driving speed (forward velocity) express time constraints and generate timed movement (m_i, n_i). Movement is obtained by integrating these dynamical systems.
Each robot forward velocity is controlled such that the vehicle has a fixed time to reach the target. The time courses of the m_i dynamical variables evolve from an initial to a final value,

yielding a timed movement with amplitude A_{ic}. The state of the movement is represented by the dynamical variable, m_i, which is not directly related to the spatial position of robot, but rather represents its temporal position. At each instant of time, the current robot position, calculated by dead-reckoning, is compared to the position each robot should have if no obstacle had been avoided. The robot velocity is dynamically controlled based on the results of this comparison: if the robot is farther from the target than what it should be, the robot is accelerated. Conversely, if the robot is closer to the target than what it should be, the robot is de-accelerated.

Target and robot position are transformed onto time-varying parameters that control the parameters of the (m_i, n_i) dynamical systems that generate timed movement in real time. More precisely, these parameters specify the amplitude A_{ic} of the movement, given by the visually detected target, current motor values and robot internal model. The inclusion of these feedback-loops enables online trajectory modulation.

The two robot movements are coupled in time such that if the two movements onsets are not perfectly simultaneous or if their time trajectories are evolving differently (one is going faster/slower than the other), leading to different movement times (time it takes to reach the target), this coupling coordinates the two movements such that they terminate approximately simultaneously.

Another velocity dynamical system assures that the system is in a stable state at all times and controls the forward robot velocity depending whether obstacles were detected or not.

The rotation speeds of both wheels are computed from the angular velocity, ω, and the forward velocity, v, of the robot. The former is obtained from the dynamics of heading direction. The later, is given by the velocity dynamics. By simple kinematics, these velocities are translated into the rotation speeds of both wheels and sent to the velocity servos of the two motors.

4.2 Attractor dynamics of heading direction

The robot action of turning is generated by letting the robot's heading direction, ϕ_h, measured relative to some allocentric reference frame, vary by making ϕ_h the behavioral variable of a dynamical system (for a full discussion see (Schoner & Dose, 1992). This behavioral variable is governed by a nonlinear vector field in which task constraints contribute independently by modelling desired behaviors (*target acquisition*) as attractors and undesired behaviours (*obstacle avoidance*) as repellers of the overall behavioural dynamics.

The direction ϕ_{tar} points towards the target location from the current vehicle position relative to the allocentric reference frame (Fig. 5). This task is expressed by a specific value of ϕ_h ($\phi_h = \phi_{tar}$). The direction ϕ_{obs} points towards the obstacle locations from the current vehicle position relative to the allocentric reference frame. The task of avoiding collisions with obstacles is expressed by the undesired behavioral state $\phi_h = \phi_{obs}$.

The specified values ϕ_{tar} and ϕ_{obs}, expressing either desired or to be avoided values for the heading direction, ϕ_h, are independent of ϕ_h since they are both measured within an allocentric reference frame. This invariance enables the design of individual behaviors independently from each other.

Integration of the *target acquisition*, $F_{tar}(\phi_h)$ and *obstacle avoidance* , $F_{obs}(\phi_h)$ contributions is achieved by adding each of them to the vector field that governs heading direction dynamics (Fig. 6)

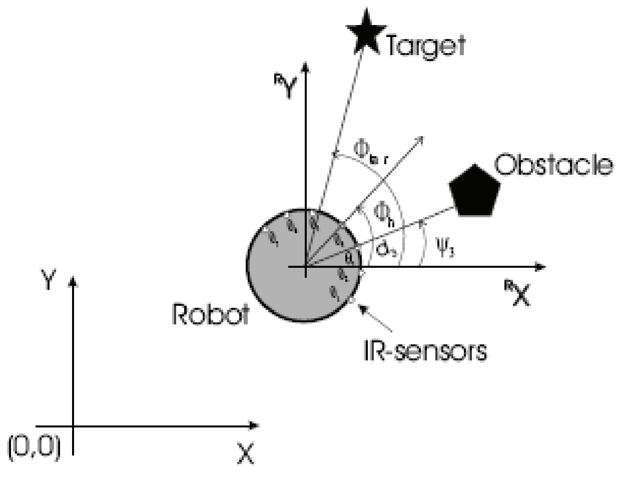

Fig. 5. The task of moving in the *(x, y)* plane toward a target while avoiding obstacles. The heading direction, ϕ_h, relative to the *x-axis* of the allocentric reference frame, is the behavioral variable which controls vehicle motion. Constraints for the dynamics of heading direction, ϕ_h, are parameterized as particular values, ϕ_{tar} and ϕ_{obs} of heading direction. These specify the directions at which target and obstacles lie from the current position of the robot. Seven Infra-red sensors are mounted on the robot's periphery at an angle θ_i relative to the robot reference frame. These sensors measure the distance d_i to objects in the direction $\psi_i = \phi_h + \theta_i$ relatively to the allocentric reference frame.

$$\frac{d(\phi_h)}{dt} = F_{obs}(\phi_h) + f_{tar}(\phi_h) + f_{stoch}(\phi_h) \tag{7}$$

We add a stochastic component force, F_{stoch}, to ensure escape from unstable states within a limited time. The complete behavioral dynamics for heading direction has been implemented and evaluated in detail on a physical mobile robot (Bicho et al., 2000; Santos, 2004).

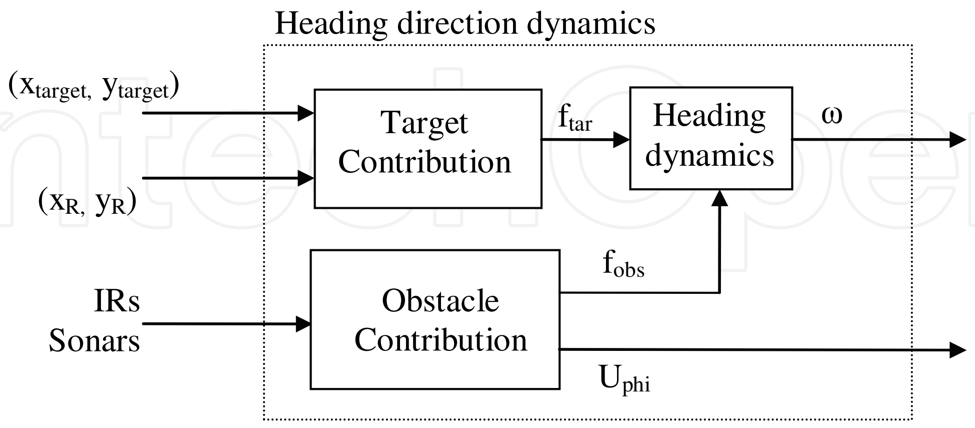

Fig. 6. Heading direction dynamics results from the sum of target and obstacle contributions. This dynamics specifies the angular velocity, ω, of the robot. By simple kinematics, angular velocity and forward velocity are translated into the rotation speeds of both wheels and sent to the velocity servos of the two motors.

4.2.1 Target acquisition

Target location, (x_{target}, y_{target}), is continuously extracted from visual segmented information acquired from the camera mounted on the top of the robot and facing in the direction of the driving speed. The angle ϕ_{tar} of the target's direction as "seen" from the robot is:

$$\phi_{tar} = \arctan \frac{y_{target} - y_R}{x_{target} - x_R} \Leftrightarrow \phi_{tar} = \arctan \frac{{}^R y_{target}}{{}^R x_{target}} \tag{8}$$

where (x_{target}, y_{target}) and (x_R, y_R) are the target location and current robot position respectively, in the allocentric coordinate system. The latter is given by the dead-reckoning mechanism. $({}^R x_{target}, {}^R y_{target})$ is the target location relatively to the robot. In a real implementation the integrated value for robot position and heading direction has some error. This error comes from the fact that motor commands are not correctly executed and the robot does not move has much as it thinks it did. Moreover, this error is cumulative during time. Thus, the position estimated for the robot as well as the estimate of the heading direction should be calibrated with respect to the external reference frame such that the robot is capable of accurately reaching the target position.
An attractive force-let is erected at the direction $\phi_h = \phi_{tar}$, specifying the position of an attractor in the heading direction dynamics:

$$f_{tar}(\phi_h) = -\lambda_{tar} \sin(\phi_h - \phi_{tar}) \tag{9}$$

This contribution is sinusoidal such as to reflect the requirement that the dynamics is the same again after the robot has made a full 360 degrees, leading to a repellor in the direction $\pi + \phi_{tar}$ opposite to ϕ_{tar}. This range expresses the fact that target acquisition behavior is desired from any starting orientation of the robot.

4.2.2 Obstacle avoidance

The robot measures distance to nearby surfaces through seven infra-red sensors mounted on a ring centred on the robot's rotation axis. Each sensor is mounted at an angle θ_i relative to the frontal direction in a reference frame fixed to the robot. Hence, relatively to the allocentric reference frame, each sensor looks into a direction, $\psi_i = \phi_h + \theta_i$ (Fig. 5).
The used strategy says that if an obstruction is detected in the direction, ϕ_i, read by each of these sensors ($i = 1, ..., 7$), a virtual object was detected in that direction. A repulsive-force, $F_{obs,i}$, centred at ϕ_i is erected for each virtual object detected and summed up for the overall obstacle avoidance dynamics

$$f_{obs}(\phi_h) = \sum_{i=1}^{7} f_{obs,i}(\phi_h) = \sum_{i=1}^{7} \lambda_i (\phi_h - \psi_i) e^{\left(\frac{-(\phi_h - \psi_i)^2}{2\sigma_i^2}\right)} \tag{10}$$

Note that the obstacle avoidance term does not depend on the current heading direction, ϕ_h. Only the position of the IR sensors relative to the robot's reference frame, which is fixed and known ($\phi_h - \psi_i = -\theta_i$), is required for the heading direction dynamics. Thus, calibration of the robot is not important within this module. In fact, as described and discussed in previous work (Schoner et al., 1995; Steinhage & Schoner, 1998) using an external reference frame for the behavior variables does not always imply a calibration of the planning coordinate system.

The two parameters within this equation that have to be mathematically defined are: the strength of repulsion of each repellor, λ_i, and the angular range, θ_i.
The former is a decreased function of the sensed distance, *di*:

$$\lambda_i = \beta_1 e^{(\frac{-d_i}{\beta_2})} \tag{11}$$

Objects farther than β_2 are repelled weakly than objects closer. The maximum repulsion strength of this contribution is controlled by β_1 (tuned later).
The latter, σ_i, determines the angular range over which the force-let exerts its repulsive effect:

$$\sigma_i = \arctan\left[\tan\left(\frac{\Delta\theta}{2}\right) + \frac{R_{robot}}{R_{robot} + d_i}\right] \tag{12}$$

The first term reflects the fact that infra-red sensors cannot determine the exact position of an obstacle within their angular range: an obstacle is assumed to cover the entire sensor sector, $\Delta\theta$ (= 30 degrees).
The second term expresses the fact that a bigger robot needs a larger distance to turn away from an obstacle that occupies maximally the entire sensor range than a smaller robot.

4.3 Coupling to sensorial information

Object tracking is a crucial research issue in robot vision, especially for the applications where the environment is in continuous changing, like mobile robot navigation, and in applications that must deal with unstable grasps (Pressigout & Marchand, 2005; Taylor & Kleeman 2003).
The most common approaches for object tracking are based on the detection of one of these three cues: edges, color and texture (Pressigout & Marchand, 2005; Taylor & Kleeman 2003) Everingham & Thomas, 2001; Zhao & Tao, 2005; Yilmaz et al., 2004).
The first concerns the extraction of a number of features of the object, like points, lines, distances and models of the contours. These features allow to have fast tracking process and also to estimate the pose of the object. Therefore, this approach is also used in visual servoing systems (Pressigout & Marchand, 2005; Armstrong & Zisserman, 1995). The fact that this is generally based on the analysis of the gradients intensity, other approaches are necessary for applications with highly textured environments or objects Pressigout & Marchand, 2005; Shahrokni et al., 2004; Yilmaz et al., 2004). For applications where the light conditions are not stable or its interaction with the objects produces shadows, the edge based techniques are not suitable as well.
When color is the main different characteristic of the object in relation with the environment, than the most suitable approaches are based on this feature. Several works can be found in the literature regarding the extraction of several characteristics based on different color spaces (Zhao & Tao, 2005; Yilmaz et al., 2004; Bradski,1998) Some works have been proved to be efficient for situations where the light conditions are not uniform and are changing during the tracking procedure (Yilmaz et al., 2004; Bradski,1998). Nevertheless, the majority of these algorithms are too computationally complex due to the use of color correlation, blob analysis and region growing.
In the presence of highly textured objects and clutter, which produce too many irrelevant edges, texture segmentation techniques are recently been used. However, because texture

segmentation techniques require computing statistics over image patches, they tend to be computationally intensive and have therefore not been felt to be suitable for such purposes (Giebel et al., 2004; Shahrokni et al., 2004; Everingham & Thomas, 2001).
However, robots cannot rely on the regular presence of distinctive colours, high contrast backgrounds or easily detected textures when tracking arbitrary objects in an unstructured domestic environment. As a result, individual cues only provide robust tracking under limited conditions as they fail to catch variations like changes of orientation and shape. Nevertheless, if flexibility and/or simplicity, speed and robustness are required they are a good option.
In this particular application the goal is to robustly detect a color target in an unstructured, complex environment. Target position is acquired by simulating a camera mounted on the top of the robot and facing in the direction of the driving speed. We have assumed that target size is known and can be measured in the image.
In our application, we have to deal with the following main computer-vision problems: (1) a clutter environment, including non-uniform light conditions and different objects with the same color pattern (distractors); (2) irregular object motion due to perspective-induced motion irregularities; (3) image noise and (4) a real-time performance application with high processing time. Some of these constraints may not be a problem in a simulated environment, but they will be as soon as we move on to a real application.
The overall vision module showing outputs and information flow is depicted in fig. 7.

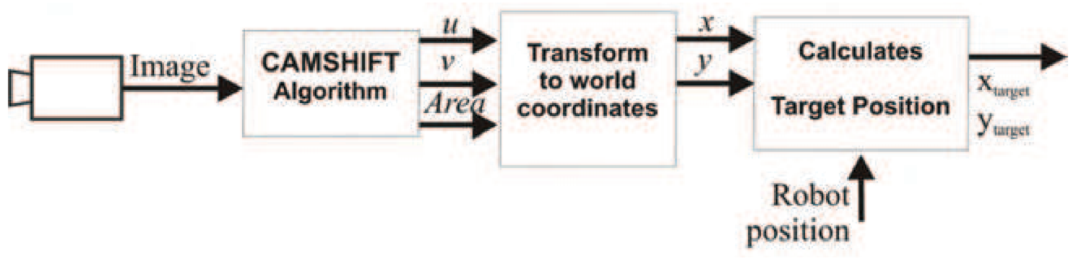

Fig. 7. Vision module.

Since the application demands a fast algorithm for color tracking, we have chosen a color based real-time tracker, Continuously Adaptive Mean Shift (CAMSHIFT) algorithm (Bradski, 1998). This algorithm deals with the described computer-vision application problems during its operation and has low computational cost.
CAMSHIFT algorithm uses a search window to track the moving object, ignoring objects outside this search window. Also scales the search window to object size thus allowing different distances between the object and the camera. The color space used is the HSV, which is less sensitive to lighting changes. The color model is mainly based on the hue histogram. The block diagram of the CAMSHIFT algorithm is presented in fig. 8 and tracks the u, v coordinates and Area of the color blob representing the target.
The CAMSHIFT tracks objects using a probability distribution image of the desired color. To do this, first, a model of the desired hue (H of the target) must be created using a 1D color histogram. During tracking, this H histogram is used as a lookup table to convert the pixels of a new image to a corresponding probability of target image. The centre and size of the color object are found via the convergence algorithm operating on the color probability image. The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image. The process is then repeated for continuous tracking. The calculus of color probability distribution for the new image will

be restricted to a smaller image region surrounding the current CAMSHIFT window. This results in large computational savings.

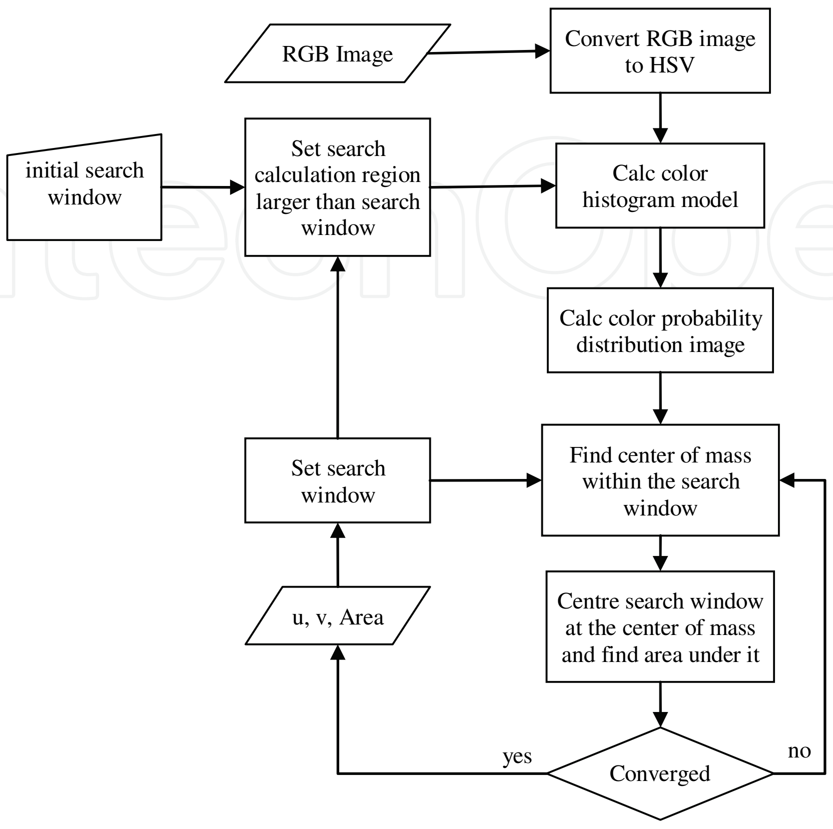
RGB Image
Convert RGB image to HSV
initial search window
Set search calculation region larger than search window
Calc color histogram model
Calc color probability distribution image
Set search window
Find center of mass within the search window
u, v, Area
Centre search window at the center of mass and find area under it
yes
no
Converged

Fig. 8. Block diagram of CAMSHIFT algorithm.

The HSV color space presents some problems, specifically when brightness is low. In this case, saturation is also low and then hue values become instable. To overcome this problem, hue that has very low corresponding brightness values must be ignored. For very low saturation values, hue is not defined and the corresponding pixels must be ignored.
CAMSHIFT tracks the u, v image coordinates and the area of the color blob representing the object. To convert the blob coordinates in pixels to camera coordinates, the camera Pinhole model was used, which estimates the rays going from point C (projection center) through the image point $m=(u,v)$ and through the world point M=(X,Y,Z), fig 9.
In this model, for a still image, a world point (X,Y,Z) in the camera frame is projected to an image point (u,v), which can be obtained using the perspective transformation as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{13}$$

where $x' = X/Z$, $y' = Y/Z$; (c_x,c_y) are the u and v coordinates of the principal point C in pixel units (optical center of the image, fig. 9) and f_x, f_y are focal lengths expressed in pixel units. $f_x=f/P_x$ and $fy=f/P_y$, where P_x and P_y are the width and height of the pixels. s factor reflects the pixel drift of the rectangular form. For most cameras the pixels are almost perfectly rectangular and thus s is very close to zero.

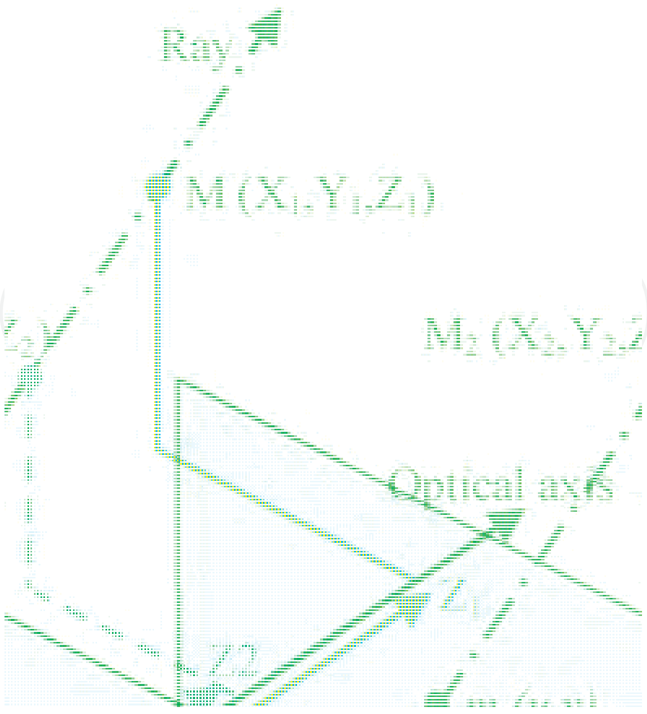

Fig. 9. Camera Pinhole representation, to estimates the rays going from point C (projection center) through the image point *m*=(*u,v*) and through the world point M=(X,Y,Z).

The matrix defined by these parameters is called the camera matrix, or the matrix of intrinsic parameters. In order to have in consideration the radial and tangential distortion of the real lens, the model is extended as follows:

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \tag{14}$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' \tag{15}$$

$$r^2 = x'^2 + y'^2 \tag{16}$$

$$u = f_x x'' + s y'' + c_x \tag{17}$$

$$v = f_y y'' + c_y \tag{18}$$

where k_1, k_2 and k_3 are radial distortion coefficients and p_1, p_2 are tangential distortion coefficients. These coefficients do not depend on the scene viewed, thus they are also considered as intrinsic camera parameters.
The Vision module calculates the world *X* and *Y* coordinates in the camera frame in mm for image points of the target centre using the distance of *Z* value, which is proportional to the Area given by the CAMSHIFT algorithm. The coordinates in the camera frame are transformed to the allocentric reference frame by a simple rotation around the X axis.
The same camera model was used for the simulation in Webots (Michel, 2004), but with different intrinsic parameters. In Webots, an ideal camera was considered.

Herein, we illustrate two real applications of this algorithm to a real, clutter environment. Fig.10a shows the result of this algorithm in the presence of a distractor element. In Fig. 10b the incident illumination as been increased by a factor of 1.5. In both situations, the algorithm is able to track the target.

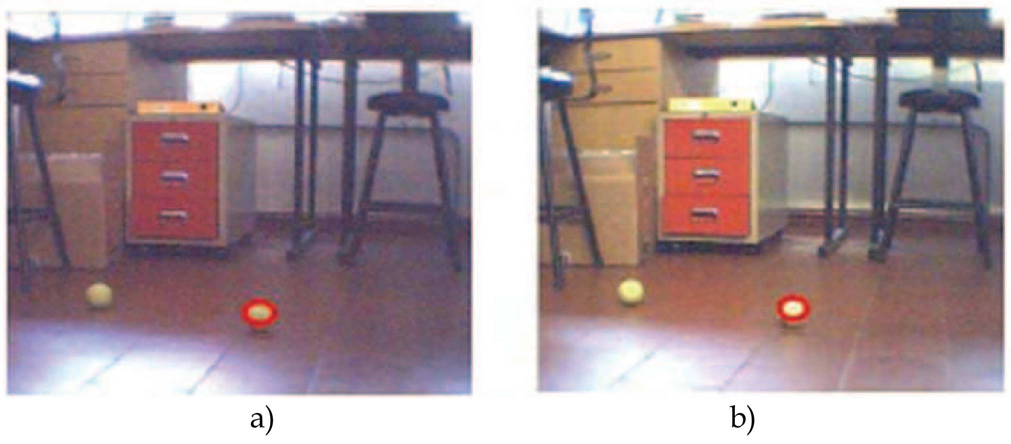

Fig. 10. Application of the CAMSHIFT algorithm to real, clutter environment. a) Presence of a distractor element. b) variations in lighting conditions.

To simulate sensor noise (which can be substantial if such optical measures are extracted from image sequences), we added either white or colored noise to the image coordinates. Here we show simulations that used coloured noise, ζ, generated from

$$\dot{\varsigma} = -\frac{1}{\tau_{corr}}\varsigma + \sqrt{Q}gwn \tag{19}$$

where *gwn* is gaussian white noise with zero mean and unit variance, so that $Q = 5$ is the effective variance. The correlation time, τ_{corr}, was chosen as *0.2 s*.

4.4 Behavior specifications

Herein, the time, *t*, visual acquired sensory information, proprioceptive data and the robot internal model, fully control the neural dynamics through the quasi-boolean parameters. The sequence of neural switches is generated by translating sensory conditions and logical constraints into values for these parameters.

The competitive advantage of the initial postural state is controlled by the parameter b_{init}. This parameter must be "on" (= 1) when either of the following is true: (1) time, *t*, is bellow the initial time, t_{init}, set by the user ($t < t_{init}$); (2) dynamical variable m_i is close to the initial state 0 ($b_{m_i colosem_{init}}(m_i)$ and time exceeds t_{init} ($t > t_{init}$) and target has not been reached.

We consider that the target has not been reached when the distance, d_{tar}, from the actual robot position (as internally calculated through dead-reckoning) and the (x_{target}, y_{target}) position is higher than a specified value, d_{margin}. This logical condition is expressed by the quasi-boolean factor, $b_{m_i hasnotreachedtarget}(d_{tar}) = \sigma(d_{tar} - d_{margin})$, where $\sigma(.)$ is a sigmoid function that ranges from 0 for negative argument to 1 for positive argument, chosen here as

$$\sigma(x) = [\tanh(10x) + 1]/2 \tag{20}$$

although any other functional form will work as well. Note that this switch is driven from the sensed actual position of the robot.
The factor $b_{m_i colosem_{init}}(m_i) = \sigma(0.15A_{ic} - m_i)$ has values close to one while the dynamical variable m_i is bellow $0.15A_{ic}$ and switches to values close to zero elsewhere. A_{ic} is the final postural state, but it is also the periodic motion amplitude which is updated based on the robot internal model, visual sensory information and the system internal state.
An additional problem arises here, however. Consider that the target position in i coordinate system (x or y) is close to robot position previous to movement initiation. In such case, a small amplitude for the corresponding periodic motion results and is dominated by $((x_R - x_{Rinit}) - m_x)$, meaning that the attractor shifts randomly. This is specially notorious when the system is in the initial postural state. The main drawback is that the periodic motion is not deactivated since the logical condition to do it is dependent on the percentage of a small value. The proposed solution is to algorithmically turn off the i periodic motion update, A_{ic}, once this changes sign relatively to the previous update and the corresponding dynamical system is in the initial postural state. Another possible solution is to replace the criterion used on $b_{m_i colosem_{init}}$ logical condition by a criterion based on absolute distances to the initial postural state.
These logical conditions are expressed through the mathematical function:

$$b_{init} = 1 - \left\{(t \geq t_{init})\left[1 - \left(b_{m_i \ close \, m_{init}}(m_i)(t \geq t_{init})b_{m_i \ has \, not \, reached \, target}(d_{tar})\right)\right]\right\} \tag{21}$$

A similar analysis derives the b_{hopf} parameter which controls the competitive advantage of the oscillatory state. b_{hopf} parameter must be on (= 1) when none of the following is false: (1) time, t, exceeds t_{init} ($t \geq t_{init}$); (2.) dynamical variable m_i is not close to the final postural state $A_{ic}.(b_{m_i notcolosem_{final}}(m_i))$; and target has not been reached $(b_{m_i hasnotreachedtarget}(d))$; and the update of the i periodic motion has not been algorithmically turned off $(b_{updateA_{ic}})$.
The factor $b_{m_i notcolosem_{final}}(m_i) = \sigma(d_{switch} - d_{crit})$ is specified based on absolute values, where d_{switch} represents the distance between m_i and the final postural state, A_{ic}. If the distance, d_{switch}, is bellow a specified value, d_{crit}, which is tuned empirically, this factor has values close to one. If this factor was not specified based on an absolute distance to the final postural state but rather was defined based on a percentage of this state, such as $0.95\,A_{ic}$, the same error as that described in factor $b_{m_i colosem_{init}}(m_i)$ would apply.
The mathematical equation that yields these results is:

$$b_{hopf} = (t \geq t_{init})b_{m_i \ not \, close \, m_{final}}(m_i)b_{m_i \ has \, not \, reached \, target}(d_{tar})\sigma(b_{update \ A_{ic}}) \tag{22}$$

Analogously, b_{final}, which controls the competitive advantage of the final postural state, can be derived from a similar analysis. b_{final} parameter must be "on" (= 1) when time, t, exceeds t_{init} (t ≥ t$_{init}$) and either of the following is true: (1) dynamical variable m_i is close to the final postural state $(A_{ic}.(b_{m_i notcolosem_{final}}(m_i)))$; (2) target has been reached $(b_{m_i reachedtarget}(d))$; (3) m_i is not close to the initial postural state zero $(b_{m_i notcolosem_{final}}(m_i))$ and (4) the update of the i periodic motion has been algorithmically turned off.

$$b_{final} = (t \geq t_{init})\left[b_{m_i \, not \, close \ m_{final}}(m_i) + b_{m_i \, reached \, target}(d_{tar}) + b_{m_i \, not \, close \, m_{final}}(m_i) + \left(1 - \sigma(b_{update \ A_{ic}})\right)\right] \tag{23}$$

4.5 Velocity

The system is designed such that the planning variable is in or near a resulting attractor of the dynamical system most of the time. If we control the driving velocity, v, of the vehicle, the system is able to track the moving attractor. Robot velocity must depend on the behaviour exhibited by the robot, depending whether or not obstacles are detected for the current heading direction value.
In case an obstacle has been detected, velocity is set as V_{obs}, which is computed as a function of the current distance to the obstacle (Bicho et al., 2000) such that good tracking of the attractor's movement is achieved:

$$V_{obs} = d\,\dot{\psi}_{\max}, \tag{24}$$

where $\dot{\psi}_{\max}$ represents the maximal rate of shift of the fixed points and is a design parameter. In case no obstacle has been detected, velocity is set as V_{timing}:

$$V_{timing} = \sqrt{\dot{m}_x^2 + \dot{m}_y^2} \tag{25}$$

where m_x, m_y are given by (Eq. 1). The path velocity, V_{timing}, of the vehicle is thus controlled through the described dynamical system architecture that generates timed trajectories.
Velocity is imposed by a dynamics equal to that described by (Bicho et al., 2000).

$$\frac{dv}{dt} = -c_{obs}\left(v - V_{obs}\right)\exp\left(-\frac{(v - V_{obs})^2}{2\sigma_v^2}\right) - c_{timing}\left(v - V_{timing}\right)\exp\left(-\frac{(v - V_{timing})^2}{2\sigma_v^2}\right) \tag{26}$$

If heading direction, ϕ_h, is currently in a repulsion zone of sufficient strength, λ_i, a strong obstacle contribution is present: $c_{obs} > 0$ and $c_{timing} = 0$ is required. In such case, this potential functional has positive values. If no obstacles are present or repulsion is weak for the current heading direction value, the above potential has negative values and $c_{obs} = 0$, $c_{timing} > 0$ is required. For further details regarding this dynamics or the above equations refer to (Bicho et al., 2000).
In the following, we briefly explain the dynamic architecture behavior of each robot that generates a timed movement from the robot resting position to a target location. At $t = 0$ s the robot is resting at its initial fixed position, (x_{Rinit}, y_{Rinit}). The robot rotates in the spot in order to orient towards or look for the target direction. At time t_{init}, the quasi-boolean for motion, b_{hopf}, becomes one, triggering activation of the corresponding neuron, u_{hopf}, and timed forward movement initiation. Amplitude of periodic motion is set as the distance between the origin of the fixed frame (same as robot position before movement initiation) and target location.
During periodic movement and at each instant of time, amplitude of i periodic motion is set as the difference between the target location and the error term between the calculated robot position and the dynamical m_i variable. If positive, this error term means the robot is ahead the position it should be if no obstacle had been circumnavigated, and the vehicle must be de-accelerated. If negative, this error term means the robot is behind the position it should be if no obstacle had been circumnavigated, and the vehicle must be accelerated. Robot i velocity is controlled according to this update. However, in the presence of obstacle contributions, the obstacle term dominates and velocity is set according to the distance to the obstacle (the maximal rate of shift of the fixed point is a design parameter).

The periodic solution is deactivated again when the x vehicle position comes into the vicinity of A_{ic}, and the final postural state (which equals A_{ic}) is turned on instead (neurons $|u_{hopf,i}| = 0$; $|u_{final,i}| = 1$). At this moment in time, the i periodic motion is no longer updated. The same behavior applies for the dynamical systems defined for the y spatial coordinate.
We want to assure that the velocity is already with the desired values, (V_{timing} or V_{obs}), before the fixed points, (ϕ_{tar}, ϕ_{obs}), change again. The desired velocity, V_{obs} or V_{timing}, must be constant from the velocity dynamics point of view. Thus, the velocity dynamics must be faster than each of the individual contributions of the heading direction dynamics. Note that the velocity variable is also coupled to the timing dynamics since V_{timing} is set dependent on the current m_i dynamical variable which is in turn dependent on the heading direction variable.
The following hierarchy of relaxation rates ensures that the system relaxes to the stable solutions, obstacle avoidance has precedence over target acquisition and target achievement is performed in time

$$\tau_{v,obs} << \tau_{v,obs}, \tau_{v,timing} << \tau_{tar}, \tau_{obs} << \tau_{tar} \tag{27}$$

5. Experimental results

The dynamic architecture was simulated in Matlab/Simulink (product of the MATHWORKS company) and in webots (Michel, 2004). This simulator is based on ODE, an open source physics engine for simulating 3D rigid body dynamics. Each vehicle has seven infrared sensors equidistantly mounted on a ring on the robot's periphery, used to measure distance to surfaces at the height of the ring. The model of the robots are as close to the real robots as the simulation enable us to be. Thus, we simulate the exact kinematic equations, mass distributions, infra-red sensor and the visual system. The dynamics of heading direction, timing, competitive neural, path velocity and dead-reckoning equations are numerically integrated using the Euler method with fixed time step. The cycle time is 70 ms and MT is 10s.
The initial heading direction is 90 degrees. Forward movement initiation is triggered by an initial time set by the user, t_{init} = 3s, and not from sensed sensorial information. Sensed obstacles do not block vision. In case the target is not currently in the limited viewing angle of the camera but has been previously seen, we algorithmically update the previous target location based on dead-reckoning information.
The rotation speeds of both wheels are computed from the angular velocity, w, and the path velocity, v of the robot. The former is obtained from the dynamics of heading direction. The later, as obtained from the velocity dynamics is specified either by obstacle avoidance contribution or by V_{timing} (eq. 25). By simple kinematics, these velocities are translated into the rotation speeds of both wheels and sent to the velocity servos of the two motors.
In order to verify if temporal coordination among the two robot movements is achieved we have performed several simulations. Herein, due to space constraints, we illustrate two exemplary simulations.
During its path towards the target, robot 2 is faced with an obstacle which it must circumnavigate (Fig. 11). This obstacle does not interfere with the robot 1 movement towards the target. Fig. 12 illustrates the robot motions and time stamps of these trajectories. The target (ball) is depicted by a light circle. Small crosses around ball position indicate ball

position as acquired by the vision systems. The robot paths are indicated by lines formed by crosses. The interval between two consecutive crosses indicates the robot's path velocity since the time acquisition interval is constant: the smaller the velocity the closer the points. When the obstacle is no longer detected for the current heading direction, at $t = 9.1s$, robot 2 is strongly accelerated in order to compensate for the object circumnavigation.

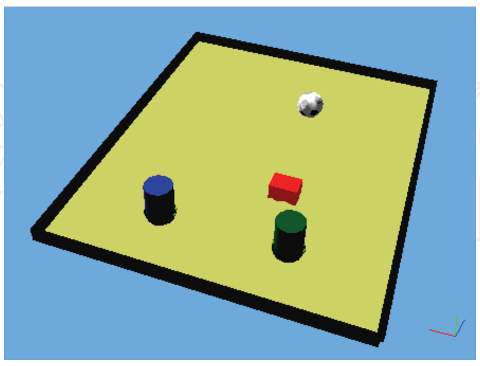

Fig. 11. Webots scenario of experiment 1.

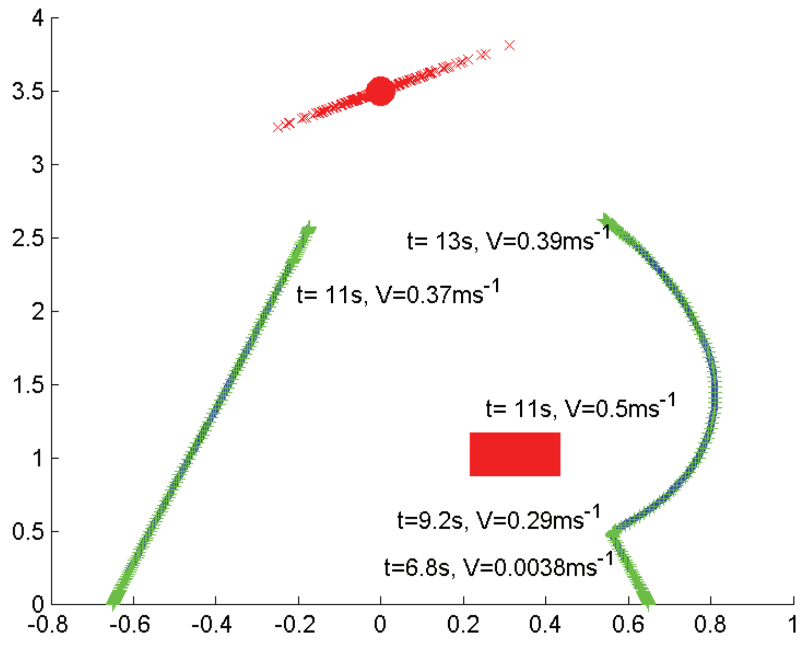

Fig. 12. The simulated robot timed trajectories for reaching the ball.

Robot velocities are depicted in Fig. 13. v represents forward velocity of the robot. v_{timing} and v_{obs} represent velocity imposed by the discussed dynamical architecture and velocity imposed in case an obstacle is detected, respectively.

The proposed dynamic architecture without coupling ($c = 0$ in eq. 2) is similar to work presented in (Santos, 2004), where results have shown that robot velocity is controlled such that the target is reached in an approximately constant time ($MT = 10s$) independently of the environment configuration and of the distance to the target.

The introduction of a coupling of this form tends to synchronize movement in the two robots. Thus, when x and/or y movement of robot 2 is affected by the environment configuration such that its periodic motion amplitude is increased, robot 1 movement is coordinated through coupling such that movements of both robots terminate simultaneously. This results in delayed simultaneous switch, around $t = 12.8$ s, among Hopf and final contributions for x and y dynamical systems of both robots (see Fig. 14). Note that synchronization only exists when both dynamical systems exhibit periodic motion.

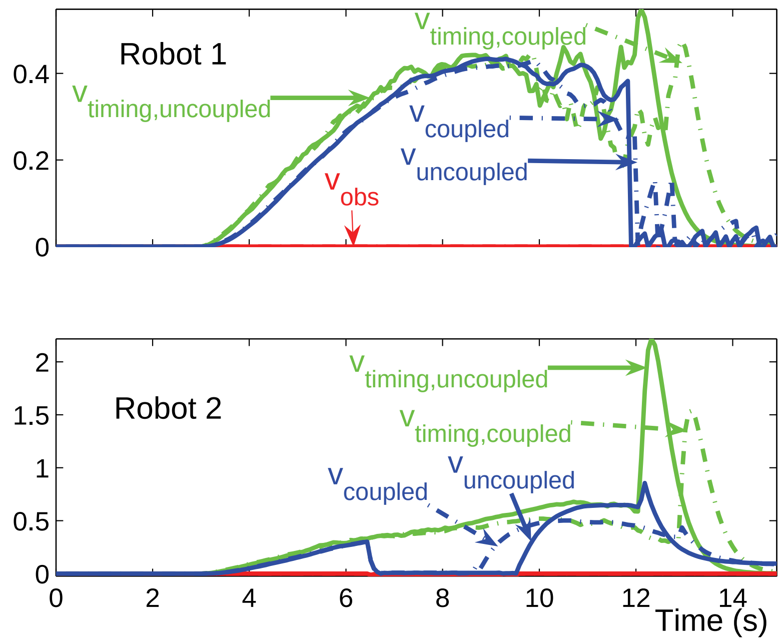

Fig. 13. Velocity variables for robot 1 and 2 for the simulation run depicted in Fig.12.

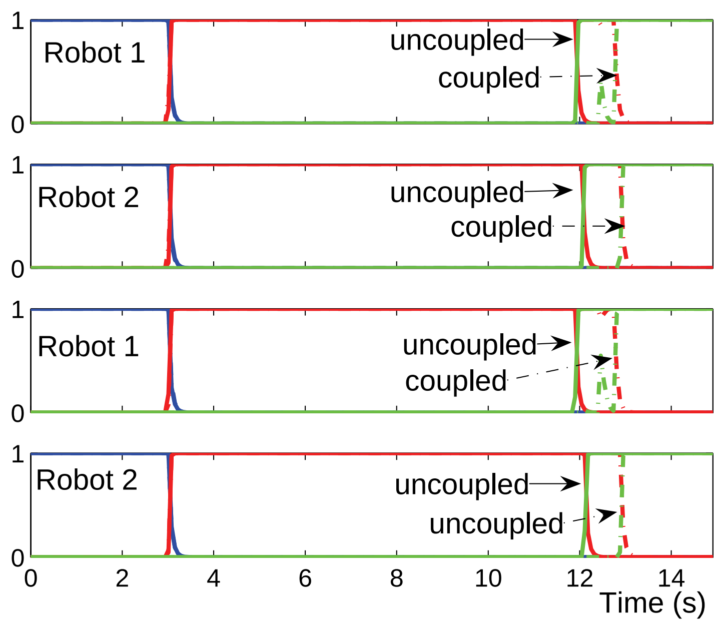

Fig. 14. u neural variables for x (top) and y (bottom) coordinate dynamical systems of both robots.

Coupling two such dynamical systems removes the need to compute exactly identical movement times for two robot movements that must be temporally coordinated. Even if there is a discrepancy in the movement time programmed by the parameter, ω, of the Hopf dynamics (which corresponds to larger *MT*s due to complex environment configurations), coupling generates identical effective movement times.

One interesting aspect is that since the velocities applied to the robots are different depending if there is coupling or not, this results in slightly different qualitative paths followed by the robot.
Fig. 15 represents the Webots scenario and fig. 16 illustrates the robot motions and time stamps of these trajectories towards the target, when both robots are faced with obstacles which they must circumnavigate. These circumnavigations lead to different movement times for both robots. Further, movements on-sets are set differently: robot 1 starts its movement at $t_{init} = 3$ s and robot 2 starts its movement at $t_{init} = 1.5$s.
The coupling coordinates the two movements such that they terminate approximately simultaneously (see Fig. 17).

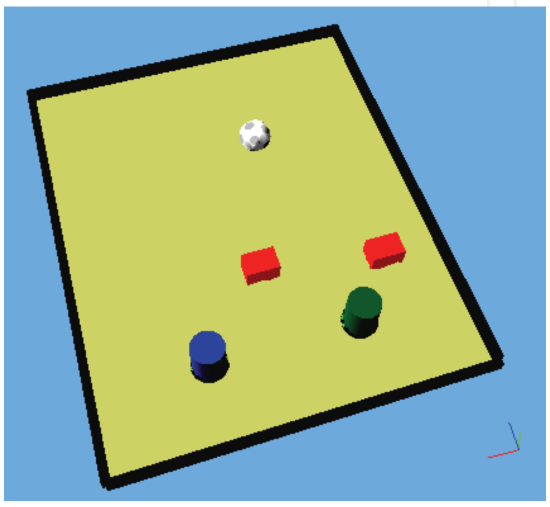
Fig. 15. Webots scenario of experiment 2.

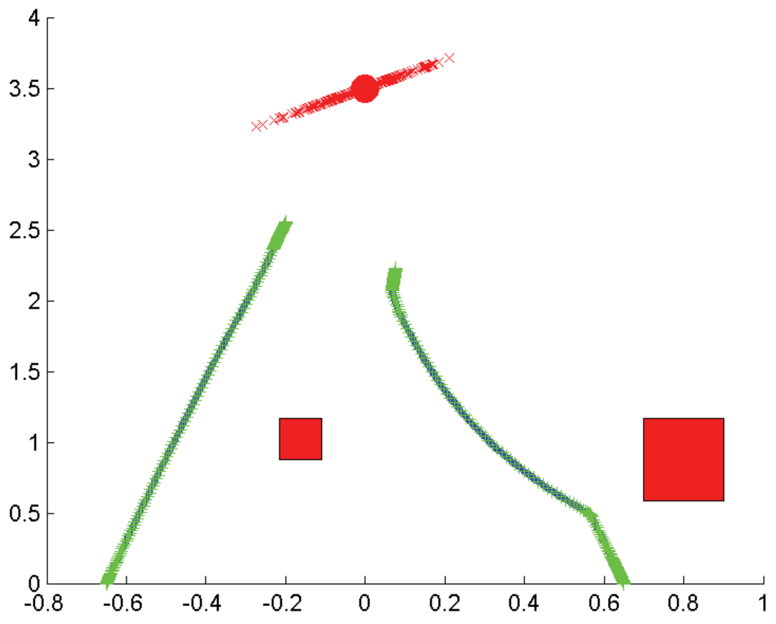

Fig. 16. A simulation run when movement on-sets are set differently for each robot. Object circumnavigation leads to different movement times for each robot.

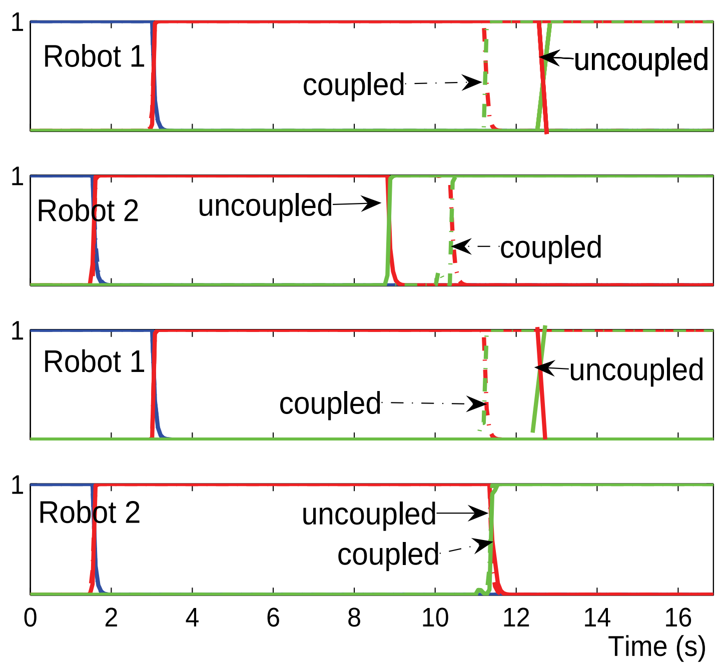

Fig. 17. u neural variables for simulation run depicted in Fig. 16.

6. Conclusion/ outlook

In this article, an attractor based dynamics autonomously generated temporally discrete and coordinated movements. The task was to temporally coordinate the timed movements of two low-level vehicles, which must navigate in a simulated non-structured environment while being capable of reaching a target within a certain time independently of the environment configuration. Movement termination was entirely sensor driven and autonomous sequence generation was stably adapted to changing unreliable simulated visual sensory information. We applied autonomous differential equations to formulate two integrated dynamical architectures which act out at the heading direction and driving speed levels of each robot. Each robot velocity is controlled by a dynamical systems architecture based on previous work (Santos, 2004), which generates timed trajectories. Temporal coordination of the two robots is enabled through the coupling among these architectures.

Results enable to positively answer to the two questions addressed in the introduction. The former asked if synchronization among two vehicles can be achieved when we apply temporal coordination among dofs. Results illustrate the dynamic architecture robustness and show that such a coupling tends to synchronize movement in the two robots, a tendency captured in terms of relative timing of robots movements.

The later question asked if the applied approach provides a theoretically based way of tuning the movement parameters such that it is possible to account for relationships among these. Results show that the coupled dynamics enable synchronization of the robots providing an independence relatively to the specification of their individual movement parameters, such as movement time, movement extent, etc. This synchronization reduces computational requirements for determining identical movement parameters across robots.

From the view point of engineering applications, the inherent advantages are huge, since the control system is released from the task of recalculating the movement parameters of the different components.
Currently, we are implementing this approach in two real wheeled vehicles which will allow to validate the proposed approach to coordination. We believe that due to the controller intrinsic properties communication issues such as delay will not be a major problem. Further, we are developing an higher dynamical system level which will implement the turning off and on of the timing control.

7. References

Armstrong, M. & Zisserman, A. (1995). Robust Object Tracking, *Proceedings of the Asian Conference on Computer Vision* , (1995).

Bicho, E.; Mallet P. & Sch¨oner,G. (2000). Target representation on an autonomous vehicle with low-level sensors, *The Int. Journal of Robotics Research*, Vol. 19, No. 5, May 2000, (424–447).

Bradski, G.R. (1998). Computer vision face tracking as a component of a perceptual user interface. *Workshop on Applications of Computer Vision*, pp. 214–219, Princeton, NJ, Oct. 1998.

Buhler, D. K. M & Kindlmann, (1994). Planning and control of a juggling robot, *International Journal of Robotics Research*, Vol. 13, No. 2, (101–118).

Erdmann, M. & Lozano-Perez, T. (1987). On multiple moving objects, *Algorithmica*, Vol. 2, (477–521).

Everingham, M. & Thomas, B. (2001). Supervised Segmentation and Tracking of Non-rigid Objects using a "Mixture of Histograms" Model. *Proceedings of the 8th IEEE International Conference on Image Processing (ICIP 2001)*, pp. 62-65, October 2001, IEEE.

Fraichard, T. (1999). Trajectory planning in a dynamic workspace: A “statetime space” approach, *Advanced Robotics*, Vol. 13, No. 1, (75–94).

Fujimura, K. & Samet, H. (1989). A hierarchical strategy for path planning among moving obstacles, *IEEE Transaction on Robotics and Automation*, Vol. 5, No. 1, February 1989, (61–69).

Fukuoka, Y.; Kimura, H. & Cohen A. (2003). Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts, *The International Journal of Robotics Research*, Vol. 3–4, (187–202).

Giebel, J.; Gavrila, D. M. & Schnörr, C. (2004). A Bayesian Framework for Multi-Cue 3D Object Tracking, *Proc. of the European Conference on Computer Vision*, Prague, Czech Republic, 2004.

Glasius, R.; Komoda, A. & Gielen, S. (1994). Population coding in a neural net for trajectory formation, *Network: Computation in Neural Systems*, Vol. 5, July 1994, (549–563).

Ijspeert, A. J.; Nakanishi, J. & Schaal, S. (2001). Learning control policies for movement imitation and movement recognition, *Neural Information Processing System (NIPS2001)*, 2001.

Large,E.; Christensen, H. & Bajcsy,R. (1999). Scaling the dynamic approach to path planning and control: Competition amoung behavioral constrains, *International Journal of Robotics Research,* Vol. 18, No. 1, (101-118).

Michel, O. (2004). Webots: Professional mobile robot simulation, *International Journal of Advanced Robotic Systems*, Vol. 1, No. 1, (39–42).

Pressigout, M. & Marchand, E. (2005). Real-time planar structure tracking for visual servoing: a contour and texture approach. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'05,* August, 2005.

Reif, J. & Sharir, M. (1985). Motion planning in the presence of moving obstacles, *Proceedings of the 25th IEEE Symposium on the Foundation of Computer Science*, pp. 144–153, Portland, OR (USA), October 1985.

Santos, C. (2005). Generating Timed Trajectories for Autonomous Robotic Platforms. A Non-Linear Dynamical Systems Approach, *Cutting edge robotics, in: Navigation Section.* (255-278).

Santos, C. (2004). Generating timed trajectories for an autonomous vehicle: A non-linear dynamical systems approach, *IEEE Int. Conf. On Robotics and Automation*. pp. 3741–3746, New Orleans, April-May 2004.

Schaal, S.; Kotosaka S. & Sternad, D. (2001). Nonlinear dynamical systems as movement primitives, *International Conference on Humanoid Robotics*, pp. 117–124, Cambridge, MA, Sept, 2001. Springer.

Schoner, G. & Santos,C. (2001). Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination, *9th Intelligent Symp. On Intelligent Robotic Systems*, pp. 18-20, Toulouse, France, July 2001.

Schoner, G. (1994). Dynamic theory of action- perception patterns: The timebefore-contact-paradigm, *Human Mov. Science*, Vol. 3, (415–439).

Schoner, G. & Dose, M. (1992). A dynamical systems approach to tasklevel system integration used to plan and control autonomous vehicle motion, *Robotics and Autonomous Systems*, Vol. 10, (253–267).

Schoner, G.; Dose, M. and Engels, C. (1995). Dynamics of behaviour: Theory and applications for autonomous robot architecture, *Robotics and Autonomous Systems*, Vol. 16, (213–245).

Shahrokni, A.; Drummond, T. & Fua, P.(2004). Texture Boundary Detection for Real-Time Tracking, *Eur. Conf. on Computer Vision,* Prague, Czech Republic, May 2004.

Steinhage, A. & Schoner, G. (1998). Dynamical system for the behavioral organization of autonomous robot navigation, *Spie-Intelligent Sys. Manufactors*, pp. 169-180, 1998.

Taylor, G. & Kleeman,L. (2003). Fusion of multimodal visual cues for model-based object tracking, *2003 Australasian Conference on Robotics and Automation*, pp. 1-3, Brisbane, December 2003.

Tani, J.; Ito, Y. & Sugita, Y. (2004). Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using rnnpb, *Neural Networls*, Vol. 17, (1273–1289). [Online]. Available: http://www.bdc.brain.riken.go.jp/ tani/publications.htm

Yilmaz, A.; Xin, L. & Shah, M. (2004). Contour-based object tracking with occlusion handling in video acquired using mobile cameras, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 26, No. 11, (1531- 1536).

Zhao, Qi & Tao, H. (2005). Object tracking using color correlogram, *IEEE Workshop on VS-PETS*, October 2005, IEEE.