Integration of vision and central pattern generator based locomotion for path planning of a non-holonomic crawling humanoid robot

Sébastien Gay, Sarah Dégallier, Ugo Pattacini, Auke Ijspeert and José Santos Victor

Abstract— In this paper we present our work on integrating a locomotion controller based on central pattern generator (CPG) and a motion planning algorithm using artificial potential fields for a non-holonomic crawling humanoid robot, the iCub. We also integrated a vision tracker and an inverse kinematics solver to perform reaching tasks. We study the influence of the various parameters of the potential field equations on the performance of the system and prove the efficiency of our framework by testing it on a physics-based robotics simulator and partially on the real iCub.

I. INTRODUCTION

Humanoids have inspired a lot of researchers and science fiction authors over the last few decades. Building a machine that would mimic humans with the same dexterity and robustness is a problem far from solved. The first step towards a fully functional humanoid robot is to enable it to move around its environment autonomously, identifying goals while avoiding collisions with obstacles.

This paper presents our work on designing a closed loop which, using only visual feedback, allows a non-holonomic humanoid robot, the iCub, to locomote in a complex environment autonomously. It uses an infant like crawling gait, and reaches targets while avoiding obstacles using a potential field based planning. As a metaphor of a real infant, one can think of a child moving around a room towards toys scattered on the ground and grab them, while avoiding to bump into the furniture.

A. Locomotion

The locomotion system we developed, already presented in [1] before, uses central pattern generators (CPG), i.e. networks of coupled oscillators inspired from the spinal cord of many animals. CPG models are increasingly used for different kinds of robots and types of locomotion such as insect like hexapods and octopods ([2]), quadrupeds ([3]), swimming ([4]), and humanoids ([5]). For a more complete review of CPGs and their application in robotics see [6]. The main benefits of CPGs for locomotion is their robustness against perturbations, the ability to smoothly modulate the shape of the oscillations with simple control signals and the possibility to easily integrate sensory feedback. Most of the efforts of the past decades have been dedicated to using CPGs for rhythmic locomotion pattern generation. Yet, periodic movements do not suit discrete tasks like manipulation or reaching. Our system embeds both rhythmic and discrete motion generation in the same CPG architecture.

B. Path planning

Numerous path planning techniques exist in the literature. Most of them use a geometric description of the environment and the robot. Grid based approaches overlay a grid on the map of the environment, reducing the path planning problem to a graph theory problem. Sampling techniques are currently considered the state of the art for a vast majority of motion planning problems. For a comparative description of grid based and sampling techniques, see [7]. Yet, both these methods require an exhaustive representation of the world to be efficient and a precise odometry estimation to be able to achieve the computed roadmap, which we do not have for our application.

Obstacle avoidance techniques are better suited to partially known environments. Examples of obstacles avoidance techniques include vector field histogram [8] which computes a subsets of motion directions and picks the best according to some heuristics and the dynamic window approach [9] which works in a similar way but in the velocity controls space.

An alternative method, at the border between path planning and obstacle avoidance techniques, is Artificial potential fields [10]. The idea is to place artificial positive potentials on obstacles and negative potentials on the goal to attain, and navigate along the gradient of the potential field. The major problem of this method is its fragility to local minima, although some harmonic potential field functions have been developed to counter this weakness [11]. This method has not been developed specifically for non-holonomic robots, and some variants based notably on fluid dynamics theory [12] have been developed to cope with the constraints of these particular robots.

We chose to use an artificial potential fields approach for our application because it is (i) easily extensible to partial descriptions of the environment and dynamically changing environments, and (ii) it is computationally inexpensive, a necessary condition for online path planning.

Our approach does not claim to design a new state of the art motion planning algorithm. Instead, the goal of this work is to study the challenges that emerge when dealing with real legged non-holonomic robots. From this perspective we have developed a framework which integrates a vision tracking system exploiting the embedded cameras of the robot, a high level motion planner based on artificial potential fields and acting on the low level CPG controller, and an inverse kinematics solver for reaching. To our best knowledge approaches integrating all these features on a

humanoid robot are very seldom in the literature. Examples include the work in [13] on the ASIMO robot which dealt with dynamical environments but where no vision was involved and a exhaustive representation of the world was provided to the robot. Other examples on different kinds of robots are found in [14] where a potential field approach was explored to plan the movement of a rover robot in an outdoor environment, and [15] which won the DARPA challenge consisting of having car robots locomote in a natural environment. This study shows that online vision based navigation can be efficient even on a legged non-holonomic robot where vision and odometry estimation are strongly perturbed by the specificities of the quadruped gait (rolling effect etc.). It also shows an application of a fully autonomous high level to low level control system based on dynamical systems allowing rhythmic (crawling) and discrete (reaching) movements. Finally one of the main concerns of this work is to match as closely as possible the constraints of the real robot. The gait that we designed implies a minimal radius of curvature of the robot when steering. The study presented here gives clues on how to adapt the parameters of the planning system to the actual constraints of the robot, when implementing on a real iCub. We also quantify the minimum radius of curvature that a robot should have to achieve acceptable performance, which could be critical information when implementing new gaits for the iCub or even when designing the next generation of the robot.

II. PRESENTATION OF THE ARCHITECTURE

A low-level controller for the generation of both discrete and rhythmic movements, based on the concept of central pattern generators (CPGs), was developed with the main focus of implementing an adaptive, closed-loop controller for crawling, in the framework of the RobotCub ([16] project. In this article, we combine this low-level architecture with a high-level planner algorithm.

After a brief description of the general hardware and software infrastructure of the iCub, we present the low-level control and then discuss more in details the high-level planner that we developed. For more information on the low-level architecture, please refer to [1] and [17].

A. Hardware and software platform

The iCub is a humanoid robot developed as part of the RobotCub project [16]. It has been designed to mimic the size and weight of a three and a half years old child (approximately 1m tall). It has 53 degrees of freedom. The iCub's eyes have 2 DOF each and are composed of two *Dragonfly 2* cameras with a 640x480 CCD sensor. The head of the robot embeds a Pentium CPU, allowing for fully autonomous control, and more demanding computation can happen outside the robot via Ethernet communication. All software modules of the iCub architecture are independent and can be distributed over a cluster of computers.

We used the Webots [18] robotics simulator, which is based on the Open Dynamics Engine for the physics simulation and on OpenGL for the rendering. It is rather realistic in the sense that is enables to set robot specific constraints such as joints limits of position, velocity, acceleration and force, as well as the proportional term P of the low level controller. Parameter of the environment like gravity, friction coefficient etc. are also open. The Webots model of the iCub fully respects the Daenavit-Hartenberg parameters of the real robot as well as the limits of the joints. The same controller is used in Webots and on the real robot thanks to a common interface.

B. Locomotion

Our locomotion framework is built on the concept of central pattern generators (CPGs), that we take in the sense of a network of unit generators (UGs) of basic movements called motor primitives ([1]).

All trajectories (for each joint) are generated through a unique set of differential equations, which is designed to produce complex movements through the superimposition and sequencing of simpler motor primitives generated by rhythmic and discrete unit generators. The dynamics of the discrete movement is simply embedded into the rhythmic dynamics as an offset. These trajectories are sent as setpoints to the PID controllers of the motors. The discrete UG is modeled by the following system of equations:

$$\dot{h}_i = d(p - h_i) \tag{1}$$

$$\dot{y}_i = h_i^4 v_i \tag{2}$$

$$\dot{v}_i = p^4 \frac{-b^2}{4} (y_i - g_i) - b \, v_i. \tag{3}$$

The system is critically damped so that the output y_i of Equations 2 and 3 converges asymptotically and monotonically to a goal g_i with a speed of convergence controlled by b, whereas the speed v_i converges to zero. p and d are chosen so to ensure a bell-shaped velocity profile; h_i converges to p and is reset to zero at the end of each movement.

The rhythmic UG is modeled as Hopf oscillator with the output of the discrete system as offset:

$$\dot{x}_i = a\left(m_i - r_i^2\right)(x_i - y_i) - \omega_i z_i \tag{4}$$

$$\dot{z}_i = a\left(m_i - r_i^2\right) z_i + \omega_i (x_i - y_i) + \sum k_{ij} z_j \tag{5}$$

$$\omega_i = \frac{\omega_{down}}{e^{-f z_i} + 1} + \frac{\omega_{up}}{e^{f z_i} + 1} \tag{6}$$

where $r_i = \sqrt{(x_i - y_i)^2 + z_i^2}$. When $m_i > 0$, Equations 4 and 5 describe an Hopf oscillator whose solution x_i is a periodic signal of amplitude $\sqrt{m_i}$ and frequency ω_i with an offset given by g_i. A Hopf bifurcation occurs when $m_i < 0$ leading to a system with a globally attractive fixed point at $(g_i, 0)$. The term $\sum k_{ij} z_j$ controls the couplings with the other rhythmic UGs j; the k_{ij}'s denote the gain of the coupling between the rhythmic UGs i and j and are set here to generate a trot gait. The expression used for ω_i allows for an independent control of the speed of the ascending and descending phases of the periodic signal, which is useful for instance for adjusting the swing and stance duration in crawling ([17]).
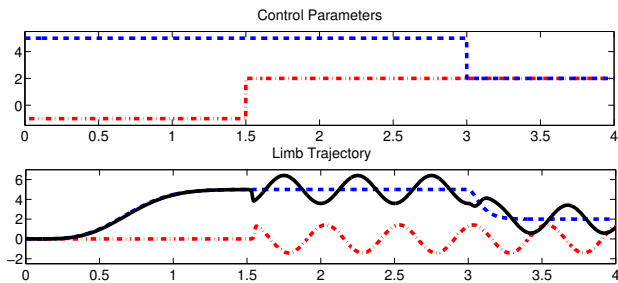
Fig. 1: **Unit pattern generators**. Upper panel. Control commands for discrete and rhythmic movements, that is the target position (in blue) and the amplitudes (in red), the frequency being not shown on the figure. Bottom Panel: The resulting discrete and rhythmic movements (resp. in blue and in red) and the trajectory embedding the two dynamics (black).

Qualitatively, by simply modifying on the fly the parameters g_i and m_i, the system can switch between purely discrete movements ($m_i < 0, g_i \neq$ cst), purely rhythmic movements ($m_i > 0, g_i =$ cst), and combinations of both ($m_i > 0, g_i \neq$ cst) as illustrated on Figure 1. Different values for the k_{ij}'s lead to different phase relationship between the limb, i.e. different gaits for instance.

C. Vision

For a robot to be able to navigate in an environment, it needs to be able to perceive it ; in our case see it. The iCub is equipped with two cameras with the same two degrees of freedom as the human eye. As visual processing is not our main topic here, we chose to use a very simple marker based tracker, based on the ARToolKit Plus library [19]. Another reason for us to use this tracker is the fact that is does not use stereo-vision to compute the three dimensional position of a fixed sized marker which for faster tracking, an especially important feature when both the eyes and the head are moving during scanning. The obstacles and the goals are marked with different markers. The tracker is able to output the 3D position in the camera reference frame and the ID of multiple markers. On the real iCub robot, the tracker is able to detect an 8cm marker and its ID about 1.5m away. It is also very robust to changes of lightning. The position of the marker is translated to the robot root reference frame (attached to the waist) using forward kinematics.

The environment that we are considering here is corridor-like and composed of goals and obstacles (See Figure 2). We placed different markers on the goals and on the obstacles. We chose a corridor-like environment so that most of the obstacles and goals appear in the field of view of the robot during locomotion. To compare the performance of our planning algorithm with different parameters, we also wanted to have a narrow environment in order to prevent the robot from turning back and have a finite dimension to have an upper bound of the performance.

D. Reaching

Once the robot has detected a goal using the vision tracker described before, it has to reach it with its hand. While approaching the goal, the robot follows it with its head and eyes to keep it in the center of its vision field. This will allow him to make sure it does not loose the goal and to have a better precision on its position. The goal position is estimated using the vision tracker described in the previous section. Once the robot reaches a specific distance to the goal, it is considered "potentially reachable". Starting from this point we use inverse kinematics to compute the joints angle of the 7-DOFs arm to achieve the target position, that is the position of the goal.

An inverse kinematics cartesian solver, (iKin) was designed specifically for the YARP framework. This solver is based on the IPOPT (Interior Point OPTimizer) library [20], a library for large scale non-linear optimization. For our problem, given a desired position x_d in $\mathbb{R}^3$, the solver finds the joint configuration q in the 7 dimensional joint space $Q \in \mathbb{R}^7$ that achieves the nearest position $K_x(\boldsymbol{q})$ of the end effector (here the hand of the robot):

$$\boldsymbol{q} = \underset{\boldsymbol{q} \in \mathbb{R}^7}{\operatorname{argmin}}(||x_d - K_x(\boldsymbol{q})||^2)$$
$$\text{s.t. } \boldsymbol{q_L} < \boldsymbol{q} < \boldsymbol{q_U} \tag{7}$$

where q_L and q_U are the lower and upper joint limits of the arm of the robot. For more details about IPOPT and the non-linear solver see [20].

It is then possible to compute the Euclidean distance between the desired and achieved positions $\rho(x, x_d)$ and set a threshold ϵ defining the reachability of the goal. Once the goal is "reachable", moves its hand to the computed position q that achieves x using the discrete system described in Section II-B.

E. Planning

The purpose of the planning module is to have the robot navigate in a world composed of multiple goals and multiple obstacles. The input of this module is a set of 3D positions of goals and obstacles sent by the vision tracker described in Section II-C and expressed in robot coordinates. The field of view of the cameras of the iCub is relatively small ($\alpha \approx 45°$) which gives a very small amount of information to the robot about its surroundings. To counter this limitation, we make the robot scan the environment by rotating its head and eyes from left to right. An egocentric partial map of the environment is built by merging the areas scanned over a full rotation of the head. The head oscillations are coupled with the limbs movements to have the scanning speed depend on the locomotion speed. The frequency of the head oscillations was set to half that of the limbs (one head rotation every two steps). This scanning made it possible to extend the vision field of the robot to $\theta \approx 120°$ (see Figure 3).

Every time a head scanning is finished, a partial map of the environment is generated and, attractive potentials $U_a(\boldsymbol{p})$ are placed on the goals and repulsive ones $U_r(\boldsymbol{p})$ on the obstacles, $\boldsymbol{p}$ being the robot 2D position on the map (note that since the map is an egocentric one, $\boldsymbol{p} = (0,0)$). Figure 2 shows an example of a partial map of the environment and the potential field associated to it.

Usual potential methods have a unique goal and thus define the attractive potential and the corresponding attracting
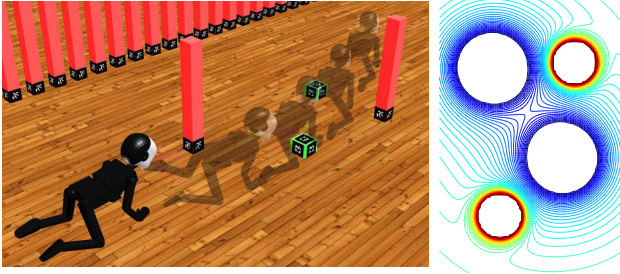
Fig. 2: Snapshot of a Webots world with the iCub at $t = t_0, t_1, t_2, t_3, t_4$. The associated potential built by the robot at $t = t_0$ (in a theoretical ideal case) is represented on the right (the scales are different). The field of action of an attractive potential is wider than that of a repulsive one due to the maximum distance of action $\rho_0 = 2$ and $k_r = 4$ of the repulsive potentials (see Equation 8)

force proportional to the distance to the goal or even to a power of it. Having multiple goals, we cannot define it this way since the robot would keep oscillating between goals without ever reaching one. Instead, we chose to define the attractive and repulsive forces so as to have a higher attraction where near a goal. This way, when several goals are in the field of view of the robot, it will go to the nearest one, which will at some point exit the field of view and the robot will head toward the next nearest one. The attractive and repulsive forces ($\vec{F}_a$) and ($\vec{F}_r$) created respectively by the goal and obstacle potentials are defined as:

$$
\vec{F}_a = -\nabla U_a(\boldsymbol{p}) = -\xi \frac{1}{\rho(\boldsymbol{p})^{k_a}} \vec{u}
$$

$$
\vec{F}_r = -\nabla U_r(\boldsymbol{p}) = \begin{cases} \eta \frac{1}{\rho(\boldsymbol{p})^{k_r}} \left(\frac{1}{\rho(\boldsymbol{p})} - \frac{1}{\rho_0} \right) \vec{u} & \text{if } \rho \le \rho_0, \\ 0 & \text{if } \rho > \rho_0. \end{cases}
$$
(8)

where :

- $\rho(\boldsymbol{p})$ is the Euclidean distance between the origin of the potential and the robot.
- ρ_0 is the maximum distance of influence of a repulsive potential.
- k_a and k_r are positive factors that determine the curvature of the potential surface.
- ξ and η are positive scaling factors.
- $\vec{u} = \nabla \rho(\boldsymbol{p})$ is a unit vector oriented away the origin of the potential and towards the robot.

The resulting force $\vec{F}_\Sigma$ that applies on the robot is then simply:

$$
\vec{F}_\Sigma = \sum_{i=0}^{n} \vec{F}_{a_i} + \sum_{j=0}^{m} \vec{F}_{r_j}
$$
(9)

n being the number of goals and m the number of obstacles.

The robot moves then of a small distance following $\vec{F}_\Sigma$. Its displacement $\vec{D}$ and angle of rotation ϕ can be defined as :

$$
\vec{D} = \Delta \frac{\vec{F}_\Sigma}{||\vec{F}_\Sigma||}
$$

$$
\phi = atan2(\vec{r}^\perp . \vec{u}, \vec{r}.\vec{u})
$$
(10)

Where where $\vec{r}$ is the current direction of motion of the robot and Δ is a small distance to be defined and $^\perp$. is the perp-dot product.
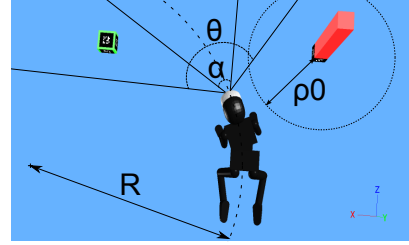

Fig. 3: A snapshot of the Webots world with annotations of the important quantities. The red pylon is an obstacle, the green cube a goal (notice the ARToolKit markers on them). α is the field of view of the robot, θ the extended field of view due to the scanning process, and R_{min} the minimum radius of curvature.

Here we only compute ϕ explicitly and let Δ be the distance achieved by the robot between two refreshs of the potential field (between two full scans). In theory, the actual rotation angle of the robot corresponds to the torso roll angle of the robot (see Figure 3), but they may somewhat different on the real robot due to sliding of the limbs on the ground.

The values of k_r, k_a and ρ_0 in Equation 8 influence strongly the shape of the potential field. Figure 4 shows this influence for two different values of ρ_0. A potential with a low k (k_r or k_a) has a slighter slope, and thus a larger range of influence than one with a big k. By varying ρ_0, one can explicitly limit the range of influence of an obstacle potential. Setting a low ρ_0 is particularly useful if one wants the robot to be able to squeeze in between obstacles. Setting a high k_r has a similar effect, while also changing the slope of the potential field. Section III presents a study of the influence of these various parameters on the performance of robots with different minimum curvature radius.
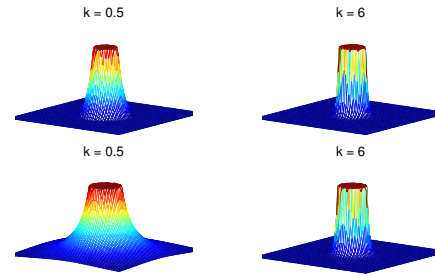

Fig. 4: Influence of k (k_r or k_a) on the shape of the potential field for $\rho_0 = 2$ (top) and $\rho_0 = 10$ (bottom). The slope of the potential surface increases with k, while ρ_0 allows an explicit limitation of the range of influence of the potential

III. RESULTS

The main questions we address here are (i) How well does our planning system perform for robots with different minimum radius of curvature R_{min}, (ii) how do the different parameters of the potential field equations described in Section II-E influence the performance and how are they related to the values of R_{min}, and (iii) what minimum value of R_{min}

should we achieve in order to reach good performance. This last point is particularly important when designing a turning gait for a robot since it helps finding a compromise between the performance of the locomotion and that of the planning. For instance, if one cannot find a stable turning gait leading to a small minimum curvature, one could decide to have the robot turn on itself by performing a series of maneuvers, at the expense of the speed of locomotion.

The performance of the system was measured by the number of reached goals versus the number of collided obstacles. In order to study the influence of the various parameters on the performance of the motion planning algorithm with the constraints of the real robot we used a two stage simulation approach: first using a 2D simulator, having enough simplicity and speed to test a wide range of parameters and then using the physics-based robotics simulator Webots ([18]). Implementation of the crawling locomotions system and the visual based reaching has been done and will be discussed at the end of this section.

A. 2D simulations

We performed a series of systematic tests using a simple 2D simulator on the following parameters: k_a, k_n, and ρ_0 and R_{min}, the minimum radius of curvature of the robot. In this simulator, no vision is involved but the field of view of the robot is constrained geometrically. Thus obstacles and goals are only "seen" by the planning algorithm if they are in an area corresponding to a field of view of $120°$, with a depth of two meters, from the robot position and along its orientation. These values are coherent with the real robot properties. We generated 70 corridor-like worlds of dimension 4×40 m, containing 10 goals and 15 obstacles each, and enclosed by walls of obstacles. The goals and obstacles were randomly positioned with the only condition that the distance between each of them was at least 1m. This is to ensure a rather uniform distribution of goals and obstacle and avoid worlds with conglomerates that would be impossible for any parameters and thus would lead to similar scores for all trials. The parameters were taken in the following sets: $k_r = \{0.5, 1, 2, 4, 6\}$, $k_a = \{0.5, 1, 2, 4, 6\}$, $\rho_0 = \{1, 1.3, 1.5, 2\}$, $R_{min} = \{0.7, 1, 2, 3, 4, 6\}$ (42000 runs). The results of these systematic tests are presented in Figure 5.

The top left graph shows the mean number of reached goals and collided obstacles over all runs for each value of R_{min}. As can be expected the smaller the minimum radius of curvature the better the performance. The fact that the number of obstacles collided is lower for $R_{min} = 6$ than for $R_{min} = 4$ is due to the corridor shape of the world tested. Indeed, for $R_{min} = 6$ the robot moves almost in straight line and thus the probability to collide with the walls is reduced. Interestingly for $R_{min} < 1$ the performance does not increase so much anymore, suggesting that a minimum radius of curvature of 1 should be sufficient to achieve near-optimal performance.
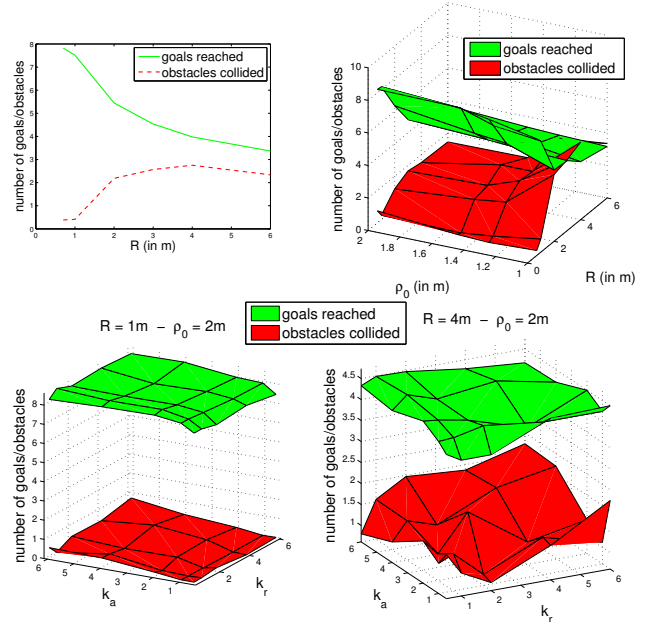

Fig. 5: Results of the systematic tests using the 2D simulator. Top left: number of reached goals and obstacles collided over the whole pool of tests. Top right: influence of ρ_0 on the performance. Bottom left: influence of k_a and k_r on the performance for a small radius of curvature ($R_{min} = 1$) for $\rho_0 = 1$. Bottom right: influence of k_a and k_r on the performance for a big radius of curvature ($R_{min} = 4$) for $\rho_0 = 2$.

The top right surface plot shows the influence of ρ_0 on the number of goals reached and obstacles collided. For a small R_{min}, the value of ρ_0 has barely any influence on the performance. The small radius of curvature of the robot allows it to avoid obstacles even if they influence its motion only very late (ρ_0 small). For higher R_{min} however, the performance strongly decreases with ρ_0. This time the robot can only avoid obstacles if it can anticipate enough (ρ_0 big).

The bottom two graphs show the influence of k_r and k_a on the performance for a small and a big value of R_{min}, and for $\rho_0 = 2$. When the radius of curvature is sufficiently small, the values of k_r and k_a are, like ρ_0 in the previous graph, not critical. This independence of the parameters for small R_{min} is a good feature of the planning algorithm for real robotics applications, since it means that the system does not significantly depend on specific parameters choices. Very small values of k_r and k_a lead to slightly lower performance, since the robot cannot get near enough obstacles to perform quick maneuvers, which would be made possible and safe by its small radius of curvature.

For big R_{min} the number of collided obstacles mostly increases with the value of k_r, since for big k_r the influence of the obstacle potentials decreases rapidly with the distance and so the robot cannot anticipate enough to cope with its big radius of curvature. A less intuitive observation is that the number of reached goals decreases for small values of k_a. This can be explained by the fact that, where several goals are in the field of view of the robot, and k_a is small, their influence would mostly balance until one is significantly nearer than the other. At that point however, with a big R_{min}, the robot would not be able to turn fast

enough to reach the nearest one. This happens in Figure 6 for $R_{min} = 4$ for the 2D simulator. At $y \approx 12$ the robot passes in between two goals without reaching any of them.
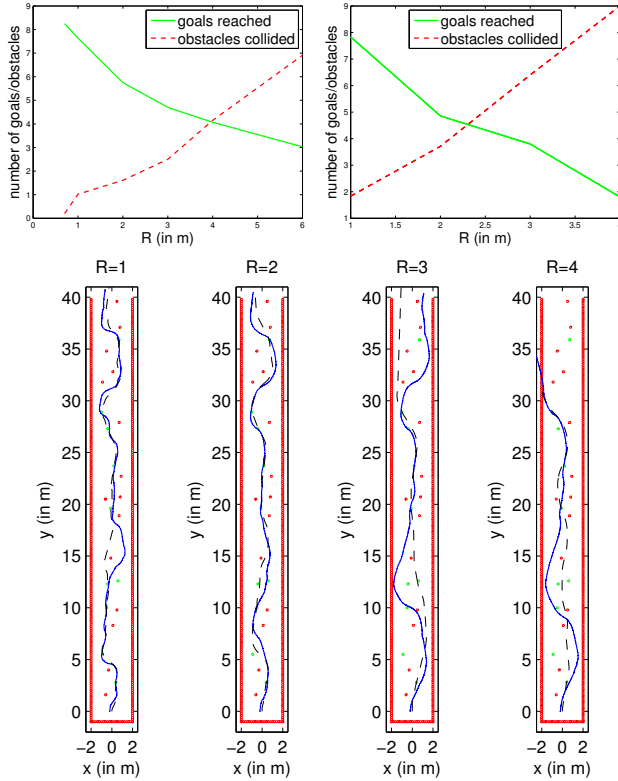

Fig. 6: Comparison of the performance of the planning algorithm for different radius of curvature in one world using the 2D simulator (top left) and Webots (top right). Comparison of trajectories with similar performance in Webots (blue solid line) and the 2D simulator (black dashed line) for different values of R_{min} (bottom figure).

B. Webots simulations

The observations made in the 2D simulator are useful to adapt a potential field based planning algorithm to the constraint of a real non-holonomic robot. But first we have to check that the behavior of a real robot would match that of the 2D simulation, at least concerning curvature radius issues. In the physics-based simulator Webots, detection of the obstacles and goals is not geometrical anymore as in the 2D simulator but uses the perspective projection Webots cameras, the "eyes" of the robot, to perform visual processing using the ARToolKit based marker tracker described in Section II-C. Hence detection is not deterministic anymore but subject to noise in the position extraction of the markers. Locomotion of course is significantly different since it uses the CPG based system described in Section II-B and not a simple translation like in the 2D simulator. This also induces noise in the vision tracking due to movements of the head and a high variance in the potential field generation since markers are constantly entering and escaping the field of view of the robot, causing the modifications in the potential field. To cope with these issues, we performed noise filtering at the vision level and introduced a short term memory at the planning level. This memory introduces damping in the

changes of the potential field and thus prevents the robot from constantly changing direction.

Due to the complexity of the simulator + locomotion + vision tracker + planning system, we only performed a limited amount of tests, to prove the efficiency of the whole framework and show that the results match that of the 2D simulation. We chose a world that gave significantly different results for different values of R_{min} in the 2D simulations. We run 5 runs for each values of $R_{min} \approx 1, 2, 3, 4$. We could not find a stable gait leading to $R_{min} < 1$ (the robot would not move) or $R_{min} > 4$ (the robot would move in straight line). A significant difference between the way the radius of convergence is computed in the 2D simulator and in Webots is worth mentioning. In Webots, turning is achieved by changing the torso roll angle (see Figure 3) and modifying the amplitudes of the left and right limbs accordingly. However, the robot cannot reach its maximum turn angle at once since it would cause a lot of sliding and big constraints on the motors. Thus at each step the turn angle increases by a small amount, and so the radius of curvature is not constant, unlike in the 2D simulator. The values of R_{min} given before are the curvature after the maximum turn angle has been reached, which may be different from the actual turn angle while navigating.

Figure 6 (top two graphs) shows the performance of the planning for different values of R_{min} in Webots and in the 2D simulator. Interestingly the relation between the maximum curvature and the number of goals reached and obstacles collided is qualitatively the same as in the 2D simulator. Thus the 2D simulator is a good approximation of the Webots simulation which should be a good approximation of the behavior of the system on the real robot. However, quantitatively, results are different, the values in the 2D simulator corresponding approximatively to those in Webots for $2 \times R_{min}$. This is mostly due to the imperfect match between the curvature in both simulators, as discussed before.

Overall the planning algorithm proved to solve well the planning problem with the proper parameters. For $R_{min} = 1$ the robot was able to reach 9 goals out of 10 and collide with no obstacle (even reach 10 in the 2D simulator). The fast online refreshing of the potential field during locomotion allows the robot to handle dynamical environments. The video attached with this paper shows the iCub navigating in a Webots world with only one goal moved around manually. The obstacles were also moved during this experiment. In the end the iCub was able follow the moving goal while avoiding the obstacles.

C. Implementation on the iCub

Finally we implemented the crawling and visual based reaching mechanisms on the real iCub robot. We did not yet implement steering and thus did not test the planning algorithm on the iCub. The experiment consisted in having the robot crawl for a couple of meters, then detect a marker placed on the ground, follow it with its head and reach it
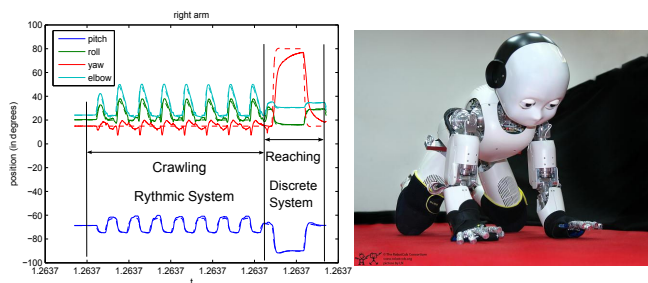
Fig. 7: Left: output of the CPG (solid line) and encoders of the four controlled joints of the right leg and right arm during crawling then reaching. Right: picture of the iCub crawling

with its right arm. Crawling proved very stable even though controlled in open loop, and the robot was able to switch instantly from rhythmic to discrete movements when reaching. Visual detection and tracking showed good performance and the robot seldom lost track of the marker before reaching it. The attached video presents this experiment. Figure 7 shows the output of the CPG and the actual trajectories of the four controlled joints of the right arm (the other limbs are qualitatively similar).

IV. CONCLUSIONS

We have presented in this document a full system to allow a humanoid to navigate in a complex environment using only vision to get knowledge about its surrounding. The low level locomotion mechanism uses coupled non-linear oscillators (CPG), to generate complex locomotion patterns using simple control inputs. This locomotion framework is able to perform rhythmic movements, for crawling, and discrete movements for reaching. At the highest level, we designed a motion planning system based on artificial potential fields and using only the visual feedback provided by a maker based tracker. The whole locomotion + vision + motion planning + reaching is thus fully autonomous.

We proved the efficiency of our system using a 2D simulator to study the influence of the various parameters of the potential field equations on the performance while respecting the constraints of non-holonomic robots and a physics-based robotics simulator to validate our system. We showed that for small radius of curvature, the system is very stable to changes in parameters, while for big radius of curvature, setting the values of k_p and k_n low and ρ_0 high allows the robot to anticipate more and compensate for its limited turning ability. These results will allow us to adapt our planning algorithm to the specificities of the turning gait when implementing on the real robot. It also gives gives us an idea of a minimum curvature radius that is necessary to attain to have good performance when navigating.

Once specificity of our work worth mentioning is the shape of the environments tested: corridor-like. We suppose that the behavior of the system would be similar in different shapes of environments since the planning system does not make any assumptions on the shape of the world and uses only local information, but proving this is left for future work.

Implementation on the real iCub showed promising results, the robot being able to crawl, track a marker on the ground

while crawling and finally reach it. Further work should include more systematic tests of the planning system in the realistic robotics environment and the implementation of a steering gait and the planning system on the real robot.

REFERENCES

[1] S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta, and A. Ijspeert, "A modular bio-inspired architecture for movement generation for the infant-like robot iCub," in *Proceedings of the 2nd IEEE RAS / EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2008.

[2] S. Inagaki, H. Yuasa, T. Suzuki, and T. Arai, "Wave cpg model for autonomous decentralized multi-legged robot: Gait generation and walking speed control," *Robotics and Autonomous Systems*, vol. 54, no. 2, pp. 118 – 126, 2006. Intelligent Autonomous Systems.

[3] H. Kimura, S. Akiyama, and K. Sakurama, "Realization of dynamic walking and running of the quadruped using neural oscillator," *Auton. Robots*, vol. 7, no. 3, pp. 247–258, 1999.

[4] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, vol. 315, no. 5817, pp. 1416–1420, 2007.

[5] G. Taga, Y. Yamaguchi, and H. Shimizu, "Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment," *Biological Cybernetics*, vol. 65, pp. 147–159, July 1991.

[6] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[7] R. Geraerts and M. H. Overmars, "Sampling techniques for probabilistic roadmap planners," *International Conference on Intelligent Autonomous Systems*, 2004.

[8] J. Borenstein, Y. Koren, and S. Member, "The vector field histogram - fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[9] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, 1997.

[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[11] K. Sato, "Collision avoidance in multi-dimensional space using laplace potential," *in Proc. 15th Conf. Robotics Soc. Jpn.*, 1987.

[12] D. Jo and K. Didier, "A reactive robot navigation system based on a fluid dynamics metaphor," in *Parallel Problem Solving from Nature*, vol. 496 of *Lecture Notes in Computer Science*, pp. 355–362, Springer Berlin / Heidelberg, 1991.

[13] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[14] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila, "Autonomous rover navigation on unknown terrains: Functions and integration," *The International Journal of Robotics Research*, vol. 21, pp. 917–942, October 2002.

[15] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Winning the darpa grand challenge," *Journal of Field Robotics*, 2006. accepted for publication.

[16] RobotCub, "http://www.robotcub.org/." An Open Framework for Research in Embodied Cognition.

[17] L. Righetti and A. J. Ijspeert, "Pattern generators with sensory feedback for the control of quadruped locomotion," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008)*, pp. 819–824, 2008.

[18] Webots, "http://www.cyberbotics.com." Commercial Mobile Robot Simulation Software.

[19] D. Wagner, "http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php." Augmented Reality Tracking Library.

[20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.