Motion Planning Using Adaptive Random Walks

Stefano Carpin Gianluigi Pillonetto[1]

***Abstract*— We propose a novel single-shot motion planning algorithm based on adaptive random walks. The proposed algorithm turns out to be simple to implement and the solution it produces can be easily and efficiently optimized. Furthermore the algorithm can incorporate adaptive components. So the developer is not required to specify all the parameters of the random distributions involved, and the algorithm itself can adapt to the environment it is moving in. Proofs of the theoretical soundness of the algorithm are provided as well as implementation details. Numerical comparisons with well known algorithms illustrate its effectiveness.**

***Index Terms*— Holonomic path planning, Adaptive sampling, Stochastic processes**

I. Introduction

In the last years the problem of robot motion planning received even more attention as a consequence of two possibly synergistic events. Starting from the seminal work presented in [1], there has been a massive introduction of randomized motion planners (see for example [2],[3],[4],[5]) which allowed to effectively face problems in high dimensional spaces, not always solvable by formerly developed planners. Moreover, motion planners are currently used for a number of applications that are beyond traditional robotics (see [6]). In fact, the continuous flow of innovative tools for efficiently searching configuration state spaces confirms that as efficient planners are introduced, the frontier of possible applications is also being pushed further. For example, we cite problems like multi-robot motion planning,

digital characters motion generation, or protein folding and ligand binding. They exhibit problem instances with tenths or even hundreds of degrees of freedom and are currently tackled by using randomized motion planners.

Recently there has also been an impulse to study algorithms not only probabilistic complete, but also resolution complete and to speculate on a possible turn back to efficient deterministic approaches ([7, 8]). However the randomized approach appears to be the current main stream of research in motion planning.

In this context we have developed a new motion planner based on random walks. In the past random walks have been sporadicly used and only in particular situations, for example to escape local minima in potential field methods ([9]). Instead, our approach relies entirely on this concept. Numerical results illustrate that difficult problems can be efficiently solved. The algorithm belongs to the class of single-shot motion planning algorithms. This means that it does not preprocess the environment to subsequently solve many queries. Instead, it is well suited for single queries. In addition, the planner is well suited for the introduction of adaptive components in order to let it adjust its random components to better address the environment where it moves.

The paper is organized as follows. Section II formally introduces the problem and illustrates the algorithm. The theoretical soundness of the algorithm is discussed in III, while section IV provides details about simulations and numerical results. Conclusions are offered in section V.

II. The Random Walk Algorithm

The problem of robot motion planning is usually dealt with by using the configuration space approach ([10]). This means that a search is conducted in the configuration space of the degrees of freedom in order to plan a motion. Every point of this space corresponds to a placement of the robot in its physical workspace. As not all the points correspond to valid placements of the robot in its workspace, the configuration space is partitioned into two subsets, the free configuration space and the obstacle configuration space. This approach has proved itself to be highly effective as it turns out that often in real applications the configuration space is a smooth manifold ([11]). Thus, it is possible to address a wide variety of problems using the same approach by designing algorithms which search for paths in these well characterized spaces.

One instance of the robot motion planning problem can be formulated as follows. Given an n-dimensional configuration space C, let C_{free} be the subset of free configurations of C. Let $x_{start} \in C_{free}$ and let X_{goal} be a target subset contained in C_{free} whose Lebesgue measure is strictly positive. The task is to find a path connecting x_{start} with X_{goal}, i.e. a continuous function $f : [0, 1] \rightarrow C_{free}$ such that $f(0) = x_{start}$ and $f(1) \in X_{goal}$. It has to be outlined that this problem belongs to the PSPACE-hard complexity class ([12]). This justifies the massive use of approximated and randomized algorithms.

Many randomized planners which have been so far developed draw independent samples from an a priori fixed probability density function whose support is the entire configuration space. The aim is then to build a graph or a tree which captures its connectivity ([1], [13]). Instead, the algorithm we propose tries to find a solution by building a random walk growing from x_{start}. At every step a new sample is generated in the neighborhood of the last point in accordance with a Gaussian distribution[2]. If the segment connecting them lies entirely in C_{free}, the point becomes the last point in the walk, otherwise it is discarded. The basic version of the algorithm is illustrated in algorithm 1. The v_k used in line 5 indicates a n-dimensional Gaussian vector with zero$-$mean vector and covariance matrix Σ_k. We also denote with "Update" the function which makes the algorithm adaptive by computing Σ_k from a certain number of last accepted samples, which in the algorithm is indicated as M. As the search of the path is carried out in a possibly high dimensional configuration space whose topological shape is usually unknown, it can be non trivial to fix the values of Σ_k. For this reason the algorithm starts with an arbitrary covariance

[1]Corresponding author. Stefano Carpin is with the School of Engineering and Science of the International University of Bremen, Germany (carpin@ieee.org). Gianluigi Pillonetto is with the Department of Information Engineering of the University of Padova, Italy (giapi@dei.unipd.it).

[2]While using both uniform and Gaussian distributions yields effective planners, we will concentrate our discussion exclusively on Gaussian distributions. This because, in addition of exhibiting better performance and being more suited for adaptivity, it allows easy and elegant proofs of the probabilistic convergence of the algorithm (see section III).

Algorithm 1 Basic Random Walk Based Motion Planner

$k \leftarrow 0$
$x_k \leftarrow x_{start}$
$\Sigma_0 \leftarrow \Sigma_{init}$
while NOT $x_k \in X_{goal}$ **do**
  Generate a new sample $s \leftarrow x_k + v_k$
  if the segment connecting x_k and s lies entirely in C_{free} **then**
    $k \leftarrow k + 1$
    $x_k \leftarrow s$
    $\Sigma_k \leftarrow$ Update$(x_k, x_{k-1}, x_{k-2}, \ldots, x_{k-H})$
  else
    discard the sample s

matrix Σ_0 and during the computation a sequence of covariance matrices is generated through the "Update" function on the basis of the evolution of the computation. For example, Σ_k could be the covariance matrix of the last H accepted samples or a similar function of the recent history of the random walk being built. As it will be shown in section III the conditions required for the probabilistic convergence of the algorithm are very mild, so that a wide variety of update rules can be used while setting the values. Specific choices will be discussed in section IV. The aim of the update step is to introduce adaptivity in the sampling distribution. In this way the algorithm is able to modulate the sampling process in order to address the specific shape of the region of the environment that the random walk is currently exploring. One important aspect of the random walk algorithm is that the time spent to generate a new sample does not depend on the size of the previously generated samples set, provided that the update of the covariance matrix can be done in constant time. This implies that the overall time spent to generate the samples is linear in the number of samples. This for example always holds if we process a fixed number of the samples accepted up to that point. This is different from most of the formerly developed algorithms, like probabilistic roadmaps (PRM) and rapidly exploring random trees (RRT), where the performance is worst than linear in the number of generated samples (see however [14] for a version of the RRT algorithm where the time needed to generate a new sample is logarithmic in the number of already generated samples). A natural technique to speed up the search process is to let the algorithm perform a bidirectional search. This means that two random walks are expanded, one growing from the start point and the other from the goal region, periodically verifying if it is possible to join them. In order to avoid the examination of all the samples generated, the connect trial is performed only between the last two generated points in the two walks. We have chosen this approach to keep constant the time needed to generate a new sample. As already observed, the expedient of bidirectional search allows a substantial gain in the performance (see [13]). Extensive experiments outlined that even this kind of bidirectional search improves the performance over the simple single direction exploration. In addition to this, further opportunism can be introduced by letting the algorithm to periodically try to connect the last sample with the goal point (or the start point if in a bidirectional search we consider the walk growing from the goal region). Another strategy we have designed to improve the performance of the algorithm consists of incorporating a *greedy* component. When the segment connecting x_k with new generated sample s does not entirely lie in C_{free} this technique is activated by trying to expand the random walk along this segment. To be specific, a set of fine grain equally spaced points $\{s_1 = x_k, \ldots, s_P = s\}$ lying on the aforementioned segment is generated. The greedy strategy then sets x_{k+1} to the point s_i such that $s_{i+1} \notin C_{free}$ and $s_j \in C_{free} \quad \forall \quad 1 \leq j \leq i$ ([13]).

When the algorithm terminates, i.e. when the last generated point is in X_{goal}, or when the two chains can be connected, the sequence of segments connecting x_i with x_{i+1} (with $i = 0, 1, \ldots$) indeed builds up a path which solves the problem. However the quality of such path is usually poor, because it includes a wide number of useless motions. Indeed the path obtained resembles a Brownian motion. A postprocessing stage is thus needed in order to smooth the generated trajectory. For this reason we used a *divide and conquer* algorithm similar to binary search (see algorithm 2, where the *pushback* operations used therein append the given point to the end of the list). See [15] for a similar approach. The smoothing procedure is repeatedly called until it is not able to further smooth the trajectory. It is worth noting that due to the good performance of the algorithm itself, the smoothing step turns out to be extremely fast and few iterative steps are needed. This will be clearly illustrated in section IV.

Algorithm 2 Solution Smoothing

SMOOTH$(V, first, last, S)$
INPUT: V vector of points to smooth
INPUT: $first, last$ extremes of V to be optimized
INPUT/OUTPUT: S list with the smoothed sequence
if $first = last$ **then**
  $S.pushback(V[first])$
else if $first = last - 1$ **then**
  $S.pushback(V[first])$
  $S.pushback(V[last])$
else if the segment connecting $V[first]$ and $V[last]$ lies entirely C_{free} **then**
  $S.pushback(V[first])$
  $S.pushback(V[last])$
else
  SMOOTH$(V, first, (first + last)/2, S)$
  SMOOTH$(V, (first + last)/2 + 1, last, S)$

III. Theoretical Foundations

In this section we provide the formalism and the proofs about the probabilistic convergence of the random walk algorithms introduced in the former section. Demonstrations will be given for what concerns the basic version, from which the convergence of bidirectional and greedy search naturally follows. See [16], [17] and [18] for similar proofs in the context of probabilistic roadmaps. After introducing the terminology used, we will define the stochastic process that will be called ARW. Then, we will show that ARW allows to move between a certain kind

of adjacent sets with a probability greater than a constant dependent only on the topology of the environment under consideration. Next, it will be shown that a path which connects the starting region with the goal region and that crosses a set more than once can be turned into a path which crosses the same set just once. These results will be then used to prove the probabilistic convergence of the new randomized motion planning algorithm.

Let Ω a probability space with $\omega \in \Omega$ ([19, 20]). Let C be $[0,1]^n$, whose generic element is denoted x and equipped with the σ-algebra $B(C)$ consisting of all the Borel sets in $\Re^n$ (i.e. the open sets and their complements as well as the countably infinite unions) contained in C. We will also denote with $\mu(A)$ the Lebesgue measure of a generic set A in $B(C)$. Moreover, we will denote $N_{x,\Sigma}(y)$ the Gaussian probability density as function of y, having mean x and covariance matrix Σ.

Definition 1: We define $B^+(C)$ as corresponding to the sets in $B(C)$ whose measure is strictly positive with respect to μ. Hereby, we will denote with C_{free} a fixed open set belonging to $B^+(C)$.

Definition 2: We call $\{x_1, x_2\}$ an admissible couple if $tx_1 + (1-t)x_2 \in C_{free}$ for $t \in [0,1]$.
Let $g : C \times C \to C$ such that:

$$\begin{cases} g(x_1,x_2) = x_1 + x_2 & \text{if } \{x_1, x_1+x_2\} \text{ is an admissible couple} \\ g(x_1,x_2) = x_1 & \text{otherwise} \end{cases}$$

We define the discrete time stochastic process $\{X_k\}_{k=0,1,2,...}$ on (Ω, Γ, η) and taking values on C, by the following recursive formulas:

$$\begin{cases} X_0(\omega) = x_{start} & \text{with } x_{start} \in C_{free} \\ X_k(\omega) = g(X_{k-1}(\omega), v_k(\omega)) & \text{for k=1,2,...} \end{cases} \quad (1)$$

where for every k the random vector $v_k(\omega)$ is normal, zero-mean with covariance matrix $\Sigma_k(\omega)$ (the dependence of a random variable on ω will be hereby implicit). We call the stochastic process in equation 1, Adaptive Random Walk (ARW).

Definition 3: We define the random vector H_k as corresponding to $\{X_0, X_1, ..., X_k\}$.

Assumption 4: For every k, v_k is independent from H_{k-1} once Σ_k is known. Moreover, every entry of Σ_k is a random variable which is measurable with respect to the σ-algebra generated by H_{k-1}, i.e. there exists for every k a function $L_k : C^k \to \mathbb{R}^{n \times n}$ such that $\Sigma_k = L_k(H_{k-1})$.

We call L_k the learning rule of the random walk at instant k.

Assumption 5: For every k, $\varepsilon_1 I_n \leq \Sigma_k \leq \varepsilon_2 I_n$ where I_n is the $n \times n$ identity matrix and $\varepsilon_1, \varepsilon_2$ are strictly positive numbers[3].

Definition 6: Given H_{k-1} and assigned $x \in C$ and $A \in B(C)$, we denote with $P^r_{k,H(k-1)}(x, A)$ the r-step transition kernel of the chain, i.e. once known the history of the chain until instant k, $P^r_{k,H(k-1)}(x, A)$ provides the probability that ARW takes value in A at instant $k + r$.
We next define the concept of visibility set ([21]).

[3]Where $A \leq B$ means that $B - A$ is positive semidefinite.

Definition 7: Assigned a set q belonging to $B^+(C)$ and contained in C_{free}, we denote with $V(q)$ the maximal open set of points $x \in C_{free}$ such that for every y belonging to q, $\{x, y\}$ is an admissible couple.

Lemma 8: Let q a set belonging to $B^+(C)$ and contained in C_{free}. Then, given an arbitrary $D \in B^+(C)$ contained in $V(q)$, there exists a strictly positive m such that for every $x \in V(q)$, k and H_{k-1}:

$$P^2_{k,H_{k-1}}(x, D) \geq m$$

Proof: By definition of $V(q)$, we have that for every $x \in V(q)$, k and H_{k-1}:

$$P^1_{k,H_{k-1}}(x, q) \geq \int_q N_{x,\Sigma_{k+1}}(y)dy \geq l\mu(q) \doteq m_1$$

where l is a real number which uniformly bounds from below the function to be integrated on q (note that the existence of l comes from Assumption 5). Moreover, we also have that for every $x \in q$, k and H_{k-1}:

$$P^1_{k,H_{k-1}}(x, D) \geq \int_D N_{x,\Sigma_{k+1}}(y)dy \geq l\mu(D) \doteq m_2$$

Combining the two inequalities, we have that for every $x \in V(q)$, k and H_{k-1}:

$$P^2_{k,H_{k-1}}(x, D) \geq m_1 m_2$$

Remark 9: It is easy to note that when $V(q)$ is convex, we could replace $P^2_{k,H_{k-1}}(x, D) \geq m$ with $P^1_{k,H_{k-1}}(x, D) \geq m$ in the statement of Lemma 8.
From this point on, if S is a subset of R^N we indicate with $\overline{S}$ the closure of the set.

Definition 10: Let T and U be subsets of C_{free}. We say that T and U are adjacent (and we write $Adj(T, U)$) when the following holds:

$$\overline{T} \cap \overline{U} \cap C_{free} \neq \emptyset$$

Lemma 11: Let A_i and A_j belong to $B^+(C)$ and to C_{free}, such that $Adj(V(A_i), V(A_j))$. Then, there exists a strictly positive m, such that for every $x \in V(A_i)$, k and H_{k-1}:

$$P^3_{k,H_{k-1}}(x, V(A_j)) \geq m$$

Proof: Since $Adj(V(A_i), V(A_j))$, there exists a point

$$x_0 \in \overline{V(A_i)} \cap \overline{V(A_j)} \cap C_{free}$$

Then, since C_{free} is open, there exists a ball $B_\varepsilon(x_0)$ with radius ε and center x_0 which is entirely in C_{free}. We define the set $D_1 = V(A_i) \cap B_\varepsilon(x_0)$. Clearly, we have $D_1 \in B^+(V(A_i))$. Then, by applying Lemma 8, there exists a strictly positive constant m_1 such that for every $x \in V(A_i)$, k and H_{k-1}:

$$P^2_{k,H_{k-1}}(x, D_1) \geq m_1$$

Let $D = V(A_j) \cap B_\varepsilon(x_0)$. Again, by applying Lemma 8 and by considering that $B_\varepsilon(x_0)$ is convex, there exists a strictly positive constant m_2, such that for every $x \in D_1$, k and H_{k-1}:

$$P^1_{k,H_{k-1}}(x, D) \geq m_2$$

Composing the two inequalities, we have that for every $x \in V(A_i)$, k and H_{k-1}:

$$P^3_{k,H_{k-1}}(x, D) \geq m_1 m_2$$

which completes the proof.

Definition 12: A triplet $\{C_{free}, x_{start}, x_{goal}\}$ is a solvable instance of the motion planning problem if the set

$$S(C_{free}, x_{start}, x_{goal}) = \{f : [0,1] \rightarrow C_{free}$$
$$\text{such that } f(0) = x_{start}, f(1) = x_{goal}, f \in C^0\}$$

is not empty.

Definition 13: Let $\{C_{free}, x_{start}, x_{goal}\}$ a solvable instance of the motion planning problem, let $f \in S(C_{free}, x_{start}, x_{goal})$ and let A be a subset of C_{free}. We say that a solution f crosses A r times if the set

$$t^f_A = \{t \in [0,1] \text{ such that } f(t) \in A \}$$

is the union of r disjoint non empty intervals (either open, closed or half open).

Lemma 14: Let $\{C_{free}, x_{start}, x_{goal}\}$ be a solvable instance of the motion planning problem. Let us suppose that there exists a finite sequence $A_1, A_2, ..., A_k$, where every A_i belongs to $B^+(C)$ and is contained in C_{free}, such that

$$C_{free} = \bigcup_{i=1}^{k} V(A_i)$$

Then, there exists $f \in S(C_{free}, x_{start}, x_{goal})$ that crosses at most once $V(A_i)$ for $i = 1, 2, ..., k$.

Proof: Let f^* be a solution of the solvable instance $(C_{free}, x_{start}, x_{goal})$ that crosses two times $V(A_j)$. Then, since $V(A_j)$ is open

$$t^{f^*}_{V(A_j)} = \{t \in [0,1] \quad | \quad f^*(t) \in V(A_j)\} = (a_1, b_1) \cup (a_2, b_2).$$

Let $p \in A_j$. Starting from f^* we define a new function f^{**} as follows

$$f^{**}(t) = \begin{cases} f^*(t) & \text{if } t \leq a_1 \\ f^*(a_1)\frac{b_2+a_1-2t}{b_2-a_1} + 2p(\frac{t-a_1}{b_2-a_1}) & \text{if } a_1 < t < \frac{a_1+b_2}{2} \\ p\frac{2b_2-2t}{b_2-a_1} + f^*(b_2)(\frac{2t-a_1-b_2}{b_2-a_1}) & \text{if } \frac{a_1+b_2}{2} \leq t < b_2 \\ f^*(t) & \text{if } t \geq b_2 \end{cases}$$

By the definition of $V(A_j)$, it follows that also f^{**} is a solution of the instance $(C_{free}, x_{start}, x_{goal})$. It is then evident that for each solution of $(C_{free}, x_{start}, x_{goal})$, by repeatedly applying the former procedure it is possible to build a new function which is also solution and indeed crosses at most once every set in the sequence $V(A_1), \ldots, V(A_k)$.

Theorem 15: Let C_{free} be connected and such that there exists a finite sequence $A_1, A_2, ..., A_k$, where every A_i belongs to $B^+(C)$, is contained in C_{free} and

$$C_{free} = \bigcup_{i=1}^{k} V(A_i)$$

Then, for each $x_{start} \in C_{free}$ and X_{goal} belonging to $B^+(C)$ and to C_{free}, the algorithm ARW started in x_{start} will reach X_{goal} with probability 1.

Proof: Without loss of generality we suppose that X_{goal} is entirely included in one of the sets of the sequence $V(A_1), V(A_2), ..., V(A_k)$ denoted $V(A_l)$. In the light of Lemma 8 and 11, there exists a strictly positive m such that for every $V(A_i)$ and $V(A_k)$ with $Adj(V(A_i), V(A_k))$ and for every $x \in V(A_i)$, k and H_{k-1}:

$$P^3_{k,H_{k-1}}(x, V(A_k)) \geq m$$

and for every $x \in V(A_l)$, k and H_{k-1}:

$$P^2_{k,H_{k-1}}(x, X_{goal}) \geq m$$

This, in addition with Theorem 14, allows us to conclude that there exists constants h and s, independent from $x \in C_{free}$, k and H_{k-1} such that

$$\sum_{r=1}^{h} P^r_{k,H_{k-1}}(x, X_{goal}) \geq s > 0$$

It follows that the probability that ARW has never entered the set X_{goal} after Zh steps is less or equal to $(1-s)^Z$. Clearly, when Z diverges, this probability goes to zero.

It is easy to observe that the last formula is similar to those found while analyzing the convergence of PRM or RRT. We outline that when dealing with the bidirectional search algorithm, as the two random walks are statistically independent, formerly introduced convergence results still hold. Moreover, for what concerns greedy search, all the lower bound inequalities are still valid, and then again the convergence holds.

Remark 16: One of the most studied aspects of stochastic processes, in particular Markov chains, concerns their behavior in the long term (see e.g. [22]). Investigations often concern the existence of a limiting distribution together with the way the process converges toward this distribution. To cite an important example, Markov chains are currently used for evaluating expectations of functions of interest under a target distribution [23]. In this context one has to design the simulation so that it can be guaranteed that asymptotically the state of the process is exactly drawn from the desired distribution. Many of these theoretical aspects related to the above defined ARW turns out to be very complex to study. This is due to the fact that we allow the transition kernel to adapt whenever new features of the configuration space are encountered during the run. As a consequence, it appears very hard to establish e.g. if ARW possesses a limiting distribution and which form it could have. In fact, the adaptation occurs infinitely often and the stationary distribution of the chain may be disturbed [24]. In the context of randomized motion planning, the interesting point is however that one

has only to design chains able to quickly visit all the free configuration space, then proving their capability to reach the goal set with probability one. This is exactly the purpose we have obtained in this Section.

IV. Simulation Details and Numerical Results

The proposed algorithm with all its variants (greedy, non-greedy, bidirectional) has been implemented and integrated into the MSL software developed at the University of Illinois ([25]), in order to compare it with different motion planning algorithms over a set of standard problems. The MSL includes a wide range of versions of RRT based motion planners as well as the basic PRM motion planner. Problems included in the MSL involve simple planar motion planning problems, as well as complicated three dimensional environments with possible multiple objects moving inside them (see figure 1 for an example). The planar problems involve 3 degrees of freedom (x,y, and the orientation), while objects in the three dimensional environments has 6 degrees of freedom (3 for the position, plus the yaw, pitch and roll angles). MSL performs collision detection using the PQP library developed at the University of North Carolina ([26]).

Fig. 1. One of the environments provided with the MSL software. The truck has to be moved out of the cage.

The adaptive rule we have used is the following (see [27]). Let

$$\overline{x}_k = \frac{1}{H} \sum_{i=k-H}^{k-1} x_i \tag{2}$$

be the average of the last H accepted samples. Given a square p-dimensional matrix M let m_{ij} be its generic element in position i, j. We define $diag(M)$ as follows

$$diag(M) = \begin{bmatrix} m_{11} & 0 & \cdots & \cdots & 0 \\ 0 & m_{22} & 0 & \cdots & 0 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 0 & \cdots & 0 & m_{p-1,p-1} & 0 \\ 0 & \cdots & \cdots & 0 & m_{pp} \end{bmatrix} \tag{3}$$

i.e. the matrix obtained from M by setting to 0 all the elements outside the main diagonal. The update rule for Σ_k is then

$$\Sigma_k = \max \left(diag \left(\frac{1}{H} \left(\sum_{i=k-H}^{k-1} x_i x_i^T - H \overline{x}_k \overline{x}_k^T \right) \right), \Sigma_{MIN} \right) \tag{4}$$

where the function *max* returns a matrix whose generic element is the greatest of the corresponding elements in the two argument matrices, and Σ_{MIN} is a diagonal constant matrix with strictly positive elements on the main diagonal. It then follows that Σ_k is diagonal too. The square roots of diagonal values of Σ_{MIN} are set to one sixth of the difference between the maximal and minimal values which can be assumed by the corresponding degree of freedom. Σ_0 is initially set to Σ_{MIN}. As we are looking for local adaptivity, i.e. short term memory, and for a fast update of the matrix Σ_k, history size is set to 10. We also have a posteriori assessed that this specific choice does not greatly influence the final results, for values of H, say less than 100. The computer used is a 2.0 Ghz Pentium IV with 512 Mbytes of RAM running Linux. All the subsequent numerical results refer to the bidirectional greedy planner which turned out to be the most efficient. Table I reports the data relative to some the standard environments provided with the MSL. For lack of space we can not give an in depth description of every environment. We however use the same names given in the MSL, so that the interested reader can refer to its documentation. Two aspects are evident. Firstly, the time spent by the algorithm is linear in the size of the generated random walk (compare columns T_1 and N_1). Secondly, even if the path produced by the random walk includes a great number of useless motions, significant improvements can be gained with the very fast post processing algorithm depicted in section II. Figure 2 illustrates an example of path smoothing. We then compare the

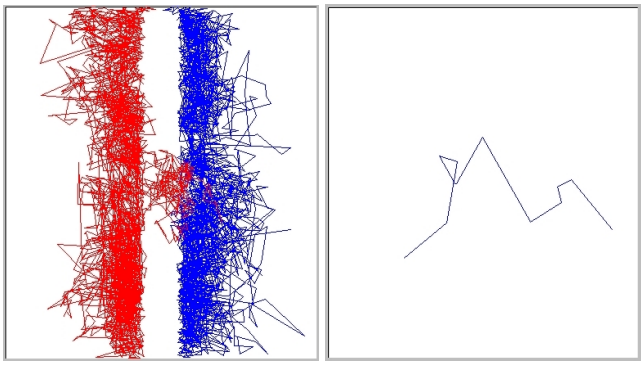

Fig. 2. The left figure illustrates a path generated using the random walk. The right figure illustrates the same path after the smoothing.

random walk algorithm performance with that concerning the basic PRM motion planner. For this aim, table II compares the overall time spent by both algorithms to find the solution. We choose to display the comparison for just the four *car like* environments where the performance of the two is comparable. In more involved problems there are usually two or three orders of magnitude of difference on the overall performance in favor of ARW based planner. It has however to be reminded that ARW is a single shot algorithm, while PRM is oriented towards repeated queries over the same environment. Finally we compare the performance of the random walk planner with the RRTConCon algorithm provided in the MSL. RRTConCon is a bidirectional single-shot greedy planner based on RRT ([13]). Of course, comparing motion planning algorithms experimentally using a restricted number of examples is hardly conclusive in general. Moreover, both algorithms have been run without

Environment	Number of points in the random walk	Time to generate the random walk	Number of points in the smoothed path	Overall Planning Time	Number of d.o.f.
Car 1	182	0.0229	10.03	0.0313	3
Car 2	907.93	0.2844	16.79	0.3497	3
Car 3	132.35	0.0372	8.9	0.0556	3
Car 4	105.7	0.0088	7.17	0.0114	3
Wrench	315.81	0.5500	23.5	1.4738	6
Cage	436.06	0.7588	14.03	1.2225	6
Truck	1731.73	7.8384	19.57	10.9321	6
Coffemug	226.78	0.3161	8.65	0.4537	6
3drigid 1	18.83	0.0532	6.22	0.1246	6
3drigid 2	8077.51	7.03	21.46	7.6171	6
3drigid 3	1657.05	1.7477	12.14	2.0408	6
Multi1	14.32	0.0181	6.43	0.057	18
Multi2	931.35	0.9464	17.6	1.3102	12
Multi3	341.2	0.5778	22.31	1.55	48
Multi6	10703.06	3.6282	69.87	4.3298	6

TABLE I

TIME SPENT IN THE VARIOUS STEPS OF THE ARW ALGORITHM. DATA ARE AVERAGED OVER 200 TRIALS AND TIME IS EXPRESSED IN SECONDS

Environment	PRM	ARW
Car 1	0.7447	0.0313
Car 2	1.3693	0.3497
Car 3	1.9961	0.0556
Car 4	0.3989	0.0114

TABLE II

COMPARISON BETWEEN PRM AND ARW PLANNERS. TIME IS EXPRESSED IN SECONDS AND DATA ARE AVERAGED OVER 200 TRIALS

Environment	RRT	ARW
Wrench	0.9824	1.4738
Cage	1.3908	1.2225
Truck	9.5486	10.9321
Coffemug	0.4825	0.4537
3drigid 1	0.0652	0.1246
3drigid 2	11.6726	7.6171
3drigid 3	14.0199	2.0408
Multi1	0.3842	0.057
Multi2	4.5345	1.3102
Multi3	84.2599	1.55
Multi6	6.3029	4.3298

TABLE III

COMPARISON BETWEEN RRTCONCON AND RANDOM WALKS

trying to find out the optimal values for the many parameters involved and this could lead to different overall performance. What emerges is however that the performance of ARW appears interesting when dealing with single-shot problems.

A. Importance of adaptivity: role of the history size

We previously anticipated that the performance of the algorithm is not very sensitive to the history size. To prove this claim, we have run the algorithm with different history sizes and compared the results (see also [28]). We also run the algorithm without adaptive components to verify the effectiveness of the proposed technique. In particular, due to its probabilistic nature, a maximum number of iterations is fixed. If the algorithm does not find a solution within that limit, this is considered a failure. Table IV illustrates the success ratio, i.e. the percentage of instances successfully solved within the given iterations bound. In all the cases the initial matrix Σ_0 was the same. Σ_0 was fixed to be again a diagonal matrix whose elements were fixed to be the square of the the difference between the maximal and minimal values which can be assumed by the corresponding degree of freedom. These values are fixed at will to be very large. It can be observed that without adaptivity the algorithm suffers from drawing samples from a bad distribution. Instead, when adaptivity is included, the covariance matrix is updated and driven to more favorable values. This is indeed a big advantage, as it this then not necessary to invest a lot of time to tune the algorithm's parameters since a rough initial choice can also be used. Table V instead compares the average time spent for different values of the history size. It can be observed that no significant variations emerge. The second important observation can be drawn from the execution times. We should remind that $H = k$ means that the last k accepted samples are used in order to compute their variance. This means that at every iteration it is necessary to process a $k \times n$ data matrix. In spite of the increased size of such matrix we can observe that the time spent is more or less constant and sometimes even decreases. Interestingly, this clearly indicates that the information acquired from the last k accepted samples is successfully used to quickly improve the sampling parameters. To make an

Environment	No Adaptivity	H=5	H=10	H=20	H=30	H=40	H=50	H=100
Coffeemug	32	100	100	100	100	100	100	100
Cage	20	100	100	99	100	100	99	99
Wrench	10	61	61	70	59	60	65	68
Truck	0	91	88	88	89	94	90	89
3drigid1	100	100	100	100	100	100	100	100
3drigid2	0	95	83	80	90	86	88	80
3drigid3	28	100	100	100	100	99	100	100

TABLE IV

SUCCESS RATIO FOR DIFFERENT VALUES OF H AND FOR THE ALGORITHM RUN WITHOUT ADAPTIVITY. PERCENTAGES ARE COMPUTED OVER 200 TRIALS FOR EACH BENCHMARK.

Environment	H=5	H=10	H=20	H=30	H=40	H=50	H=100
Coffeemug	2.089	2.118	2.208	1.988	2.169	2.231	1.978
Cage	3.463	3.219	3.690	4.128	3.476	4.085	3.935
Wrench	8.816	7.954	8.809	7.603	7.857	8.243	8.063
Truck	38.277	35.605	30.095	38.317	36.251	34.542	31.422
3drigid1	0.355	0.400	0.389	0.369	0.392	0.400	0.388
3drigid2	14.908	14.660	14.484	14.287	14.893	14.901	13.515
3drigid3	3.564	3.413	3.065	3.035	3.173	2.946	3.106

TABLE V

TIME PERFORMANCE OF THE ARW ALGORITHM FOR DIFFERENT VALUES OF THE HISTORY SIZE H. TIME IS MEASURED IN SECONDS. AVERAGE IS COMPUTED OVER 200 TRIALS FOR EACH BENCHMARK.

example, when the variance of the increments throw which the robot is currently moving is too large, so as to make many of the proposed movements fall outside the free configuration space, the algorithm is able to automatically reduce the variance of its steps, thus augmenting the probability to explore a narrow space, e.g. a corridor, in a more effective and rapid way (see figure 3. An important point which will deserve more attention

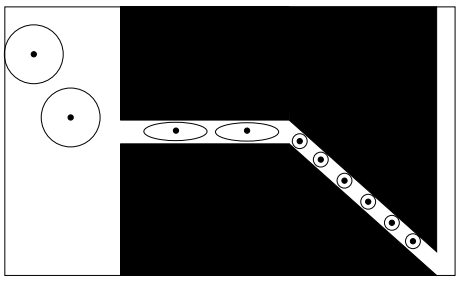

Fig. 3. When the robot moves in wide spaces (left) then a big covariance allows to try big steps. When it moves into the horizontal corridor, the covariance matrix indicates that big steps along the x axis are still possible, but not along the y direction. This because proposed samples with big displacements along the y directions are refused, and then the corresponding matrix entry reduced. In the diagonal corridor small steps in both directions will be taken, as samples with big steps in both directions will be rejected. On the other hand, if one allows for off-diagonal covariance matrix, then the covariance ellipsoid could align with the passage.

in the future is an accurate investigation in order to get clearer indications about the equilibrium point, i.e. to have indications about a range of values for H giving a minimum in the search times. This point is clearly non trivial since the expected time for finding the solution is of course a complex function of H which depends on the unknown environment where the robot moves. We also mention that we have repeated the simulations by computing the correlation between the last accepted samples but it turned out that no significant speed up can be observed by using non diagonal matrices. Of course, the design of more refined adaptivity rules could be important to further improve the performance of the proposed algorithm. For example, if knowledge about the C-space was available, it could be used in the choice of an adaptation rule for a more efficient path-finding.

V. CONCLUSIONS AND FUTURE WORK

We introduced the first random walk based adaptive motion planner. The algorithm builds a random walk with Gaussian increments over the configuration space. As the produced path can be very uneven, an efficient post processing step is used, yielding a fast smoothing of the produced path. A simple but effective technique for letting the algorithm adapt the random distribution parameters, in order to speed up the exploration process, has been devised. Thanks to this feature it is possible to get a variable sampling resolution, which is indeed believed to be one of the most promising research directions. Rigorous proofs of the theoretical soundness of the algorithm have been provided under mild assumptions on the environment to explore and on the learning rule which makes the algorithm adaptive. Numerical results clearly illustrate that for most of the investigated problems the adaptive random walk planner appears to be competitive. A direction for further developments is the introduction of a bias to let the samples be generated not only

according to the covariance matrix, but also taking into consideration unexplored regions. We are at the moment working on this extension. In the near future we also plan to investigate the use of more refined adaptive techniques which could be useful while applying the algorithm in domains different from robot motion planning. For example it appears particularly interesting to apply the algorithm to bioinformatics problems like protein folding and ligand docking. In this context, we expect the adaptivity to be even more important, thanks to the continuous nature of the energetic levels concerning proteins instead of the boolean nature of the configuration space characterizing robot motion planning. This will likely call for the study of more refined online learning techniques than those presented herein.

ACKNOWLEDGMENTS

We thank professor Steve LaValle for making freely available the MSL software. We also acknowledge the referees for constructive criticism.

REFERENCES

[1] L.E. Kavraki, P. Švestka, J.C. Latombe, and M.H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[2] R. Bohlin and L.E. Kavraki, "Path planning using lazy prm," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, May 2000, pp. 1469–1474.

[3] L.K. Dale and N.M. Amato, "Probabilistic roadmaps - putting it all together," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, May 2001, pp. 1940–1947.

[4] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Robotics : The Algorithmic Perspective.The Third Workshop on the Algorithmic Foundations of Robotics*, P.K. Agarwal, L.E. Kavraki, and M.T. Mason, Eds., pp. 142–153. A.K. Peters, 1998.

[5] P. Leven and S. Hutchinson, "Towards real-time path planning in changing environments," in *Algorithmic and Computational Robotics: New Directions*, D. Rus B. Donald, K. Lynch, Ed., pp. 363–376. A.K. Peters, 2001.

[6] J.C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *The International Journal of Robotics Research - Special Issue on Robotics at the Millennium*, vol. 18, no. 11, pp. 1119–1128, 1999.

[7] P. Cheng and S.M. LaValle, "Deterministic resolution complete rapidly-exploring random tree," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, May 2002.

[8] S. M. LaValle and M. S. Branicky, "On the relationship between classical grid search and probabilistic roadmap," in *Workshop on the Algorithmic Foundations of Robotics*, 2002.

[9] J. Barraquand and J.C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[10] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, 1983.

[11] J.C. Latombe, *Robot Motion Planning*, Kluver Academic Publishers, 1990.

[12] J.H. Reif, "Complexity of the mover's problem and generalization," in *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, 1979, pp. 421–427.

[13] J.J. Kufner and S.M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE Conference on Robotics and Automation*, San Francisco, April 2001, pp. 995–1001.

[14] A. Atramentov and S.M. LaValle, "Efficient nearest neighbor searching for motion planning," in *Proceedings of the IEEE Conference on Robotics and Automation*, Washington, May 2002, pp. 632–637.

[15] G. Sánchez and J.C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," *International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[16] L.E. Kavraki, M.N. Kolountzakis, and J.C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.

[17] D. Hsu, J-C. Latombe, and R. Motwani, "Path planning and expansive configuration spaces," *International Journal of Computational Geometry and Applications*, vol. 9, pp. 495–512, 1999.

[18] A. Ladd and L. Kavraki, "Generalizing the analysis of prm," in *Proceedings of the IEEE international conference on robotics and automation*, 2002, pp. 2120–2125.

[19] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1991.

[20] B. Øksendal, *Stochastic Differential Equations*, Springer, 1998.

[21] M. de Berg, M. van Kreveld, M.Overmars, and O. Schwarzkopf, *Computational Geometry*, Springer, 2000.

[22] S.P. Meyen and R.L. Tweedie, *Markov Chains and Stochastic Stability*, Springer-Verlag, 1993.

[23] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, Eds., *Markov Chain Monte Carlo in Practice*, Chapman & Hall, 1996.

[24] W.R. Gilks, G.O. Roberts, and S.K. Sahu, "Adaptive markov chain monte carlo through regeneration," *Journal of the American Statistical Association*, vol. 93, pp. 1045–1054, 1998.

[25] S.M. LaValle, "Msl - the motion strategy library software," http://msl.cs.uiuc.edu.

[26] E. Larsen, S. Gottschalk, M.C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Tech. Rep. TR99-018, Department of Computer Science, University of N. Carolina, Chapel Hill, 1999.

[27] H. Haario, E. Saksman, and J. Tamminen, "Adaptive proposal distribution for random walk metropolis algorithms," *Computational Statistics*, vol. 14, 1998.

[28] S. Carpin and G. Pillonetto, "Learning sample distribution for randomized robot motion planning: role of history size," in *Proceedings of the International Conference on Artificial Intelligence and Applications*, 2003, pp. 58–63.