Path Optimization for Humanoid Walk Planning: an Efficient Approach
Antonio El Khoury, Michel Taïx, Florent Lamiraux

To cite this version:

HAL Id: hal-00572375
https://hal.archives-ouvertes.fr/hal-00572375v3
Submitted on 21 Jun 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
PATH OPTIMIZATION FOR HUMANOID WALK PLANNING: AN EFFICIENT APPROACH

Antonio El Khoury\textsuperscript{1,2}, Michel Taïx\textsuperscript{1,2}, Florent Lamiraux\textsuperscript{1,2}

\textsuperscript{1}CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France
\textsuperscript{2}Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ; F-31077 Toulouse Cedex 4, France
aelkhour@laas.fr, taix@laas.fr, florent@laas.fr

Keywords: humanoid robot, motion planning, walk planning, holonomic constraints, optimization, A*, HRP-2.

Abstract: This paper deals with humanoid walk planning in cluttered environments. It presents a heuristic and efficient optimization method that takes as input a path computed for the robot bounding box, and produces a path where a discrete set of configurations is reoriented using an A* search algorithm. The resulting trajectory is realistic and time-optimal. This method is validated in various scenarios on the humanoid robot HRP-2.

1 RELATED WORK AND CONTRIBUTION

The problem of humanoid walk planning can be defined as follows: given an environment and a humanoid robot with start and goal placements, a collision-free trajectory needs to be found. It should ideally represent realistic human motion, i.e. a motion similar to that of a human being in the same conditions. This result is desirable since humanoid robots are bound to move in man-made environments such as homes, offices, and factories and because it can help them blend in among humans.

1.1 Humanoid Walk Planning

The problem of motion planning is now well formalized in robotics and several books present the various approaches (Latombe, 1991; Choset et al., 2005; LaValle, 2006). Sampling-based methods rely on random sampling in the configuration space (CS) and use for instance Probabilistic Roadmaps (PRM) (Kavraki et al., 1996) or Rapidly-Expanding Random Trees (RRT) (Kuffner and LaValle, 2000). With these methods it is possible to solve problems for systems with large numbers of Degrees of Freedom (DoF).

The motion planning problem is certainly a complex one in the case of humanoid robots, which are high-DoF redundant systems that have to verify bipedal stability constraints. Various planning strategies can be found in literature.

One category relies on whole-body task planning: kinematic redundancy is used to accomplish tasks with different orders of priorities (Khatib et al., 2004; Kanoun et al., 2009). Static balance and obstacle avoidance can thus be defined as tasks that the algorithm has to respect.

Works of (Kuffner et al., 2001; Chestnutt et al., 2005) describe in particular humanoid footstep planning schemes. Starting from an initial footstep placement, they use an A* graph search (Hart et al., 1968) to explore a discrete set of footstep transitions. The search stops when the neighborhood of the goal foot-
step placement is reached. This approach is not practical in some environments with narrow passages, and (Xia et al., 2009) reduced the computational cost of footstep planning by using an RRT planning algorithm.

Another strategy consists of dividing a high-dimensional problem into smaller problems and solving them successively (Zhang et al., 2009). The idea of dividing the problem into a two-stage scheme is described in (Yoshida et al., 2008). A 36-DOF humanoid robot is reduced to a 3-DOF bounding box. Using the robot simplified model, the PRM algorithm solves the path planning problem and generates a feasible path for the bounding box. A geometric decomposition of the path places footsteps on it, and a walk pattern generator based on (Kajita et al., 2003) finally produces the whole-body trajectory for the robot. In (Moulard et al., 2010), this two-stage approach is also used; numerical optimization of the bounding box path produces a time-optimal trajectory that is constrained by foot speed and distance to obstacles.

Another important issue is the notion of holonomic motion: while wheeled robots always remain tangent to their path, thus following a nonholonomic constraint, legged robots can also move sideways, and their motion can be described as holonomic. The path planning scheme in (Yoshida et al., 2008) is designed to this end; a PRM algorithm first builds a roadmap with Dubins curves (Dubins, 1957); but such curves impose a nonholonomic constraint and narrow passages cannot be crossed. The roadmap is therefore enriched with linear local paths. As a result this planning scheme generates motion such that the robot remains tangent to its path most of the time and uses sidestepping only in narrow passages.

Furthermore, (Mombaur et al., 2010) conducted a series of walking experiments that allowed them to build a model of human walking trajectories; if a human being walks long distances, his body tends to be tangential to his path, while holonomic motion is used over smaller distances. This is an attractive property for computed paths if a realistic motion is to be achieved, and holonomic motion can as well be used to pass through narrow spaces. These results suggest that a good combination of both nonholonomic and holonomic motions can be used to achieve realistic walking.

1.2 Contribution

The work of (Moulard et al., 2010) solves the walk planning problem in a natural way, i.e. it uses numerical optimization to minimize the robot walking along the path while following speed and obstacle distance constraints. After having tried this approach, it was empirically concluded that achieving successful numerical optimization in any kind of environment is a difficult and computationally expensive task; in fact, it requires computing a large set of parameters to fully define the optimized path.

While using the same two-stage approach of (Yoshida et al., 2008), a simpler heuristic method that generates realistic time-optimal humanoid trajectories is proposed. First the PRM algorithm and the Dubins local paths are replaced with an RRT-Connect algorithm and linear local paths. The path is then optimized by locally reorienting the robot bounding box on a discrete set of configurations of the path. Priority to nonholonomic motion is considered and holonomic motion is used only to pass in narrow passages and to avoid nearby obstacles.

The following section presents this method and explains how it is integrated in the motion planning scheme. Examples of different scenarios, including a real one with the HRP-2 platform, are shown in section 3.

2 OPTIMIZATION BY REGULAR SAMPLING

Assuming full knowledge of the environment, the RRT algorithm produces a collision-free piecewise linear path \( P_{RRT} \) for the robot bounding box (in offline mode), i.e. the path consists of the concatenation of

![Figure 2: Top view: (a) RRT-Connect path for the bounding box passing between two chairs. (b) Optimized bounding box path by random optimization (RO). (c) Optimized bounding box after adding regular sampling optimization (RSO).](image-url)
linear local paths $L_{P_{RRT}}$.

Due to the probabilistic nature of RRTs, $P_{RRT}$ may not be optimal in terms of length, and a preliminary random shortcut optimization (RO) can be run in order to shorten it (See Figure 2). While the optimized path $P$ is collision-free, the bounding box orientation is such that it could lead to an unrealistic trajectory that is not time-optimal. For instance, the humanoid robot could spend a long time walking sideways or backwards over a long distance in an open space. An additional optimization stage is introduced to address this issue in the next section.

### 2.1 Bounding Box Path Optimization

Note that each configuration $q$ can be written as $q = (X, \theta)$, where $X = (x, y)$ describes the bounding box position in the horizontal plane, and $\theta$ gives its orientation. The optimizer reorients the bounding box along $P$ by changing $\theta$ while retaining the value of $X$.

For this purpose, an A* search algorithm is executed; first $P$ is regularly sampled. Using a discrete set of possible orientations for each sample configuration and an adequate heuristic estimation function, the bounding box orientation is then modified along $P$. An optimized path $P_{opt}$ is created and leads to a realistic time-optimal trajectory.

#### 2.1.1 Preliminary Notations

After running RO on the piecewise linear path $P_{RRT}$, the path $P$ is also piecewise linear, and its first and last configurations are denoted by $q_s$ and $q_f$.

Let $d_{sample} \in \mathbb{R}^+$ be a sampling distance. Sampling $P$ with a distance $d_{sample}$ means dividing each local path $L_{P_j}$ of $P$ into smaller local paths of length $d_{sample}$; each new local path end is a sample configuration. The $n$th sample configuration of $P$ in its initial state can be obtained by indexing new local path ends starting from $q_s$, and is denoted by $q_n^{init}$.

The possible orientation states need to be defined. We aim to make a humanoid robot reach its goal as soon as possible. Since the robot is faster while walking straight than side-stepping, we attempt to change the orientation of each initial sample configuration $q_n^{init}$ such that the bounding box is tangent to the local path and introduce a new configuration denoted by $q_n^{front}$. To take into account the fact that there may be obstacles that forbid a frontal orientation, we also create $q_n^{lat_1}$ and $q_n^{lat_2}$ that are rotated by $\frac{\pi}{2}$ and $-\frac{\pi}{2}$ relative to the path tangent, see Figure 3. One particular case is local path end configurations: the mean direction of the two adjacent local paths is considered to define frontal and lateral configurations. This is done to ensure a smooth transition between two local paths.

A sample configuration whose orientation is unknown will be denoted by $q_n^{state}$. It can have any orientation state of the set $\{init, front, lat_1, lat_2\}$ except for $q_s$ and $q_f$ which remain in their initial state. Ideally, the algorithm should be able (as long as there are no obstacles) to put each sample configuration in the frontal state, create a new path $P_{opt}$, and generate a time-optimal trajectory for the robot.

An A* search is run to achieve this goal; the algorithm functions are described in the following section.

#### 2.1.2 A* Function Definition

An A* search algorithm can find an optimal path in a graph as long as a graph and an evaluation function are correctly defined. Starting from $q_s$, A* expands in each iteration the possible transitions from one sample to the next one in the graph and evaluates with the evaluation function the cost of going through each different state, see Figure 3.

A graph $G$ is defined to be a set of nodes and arcs. A valid node $q_n^{state}$ is defined to be a configuration with no collisions, and a valid arc $q_{n-1}^{state} \rightarrow q_n^{state+1}$ is a collision-free local path. The whole graph $G$ could be built before running A* by testing all nodes and arcs and making sure they are collision-free. But collision tests are slow, and A* uses a heuristic estimation function to avoid going through all nodes. An empty graph $G$ is thus initialized and nodes and arcs are built only when necessary. A successor operator needs to be defined for this purpose.

The Successor operator $\Gamma(q_n^{state})$: Its value for any node $q_n^{state}$ is a set $\{(q_{n-1}^{state+1}, c_{n,n+1})\}$, where $q_{n-1}^{state+1}$ denotes a successor node, and $c_{n,n+1}$ is the cost of going from $q_n^{state}$ to $q_{n-1}^{state+1}$. The cost $c_{n,n+1}$ is defined to be the distance $D(q_n^{state}, q_{n-1}^{state+1})$ be-

![Figure 3: Each initial sample configuration can be rotated and be in one of four states. Starting from $q_s$, the A* search algorithm searches the graph $G$ that contains only valid nodes and arcs to produce an optimized path $P_{opt}$.](image)
between two nodes of $G$, it computes the walk time from $q_{n_{\text{state}}}^n$ to $q_{n+1_{\text{state}}}$.

The speed constraint $C$ is defined as:

$$C = \begin{cases} 
\left( \frac{v_f}{v_{\text{max}}} \right)^2 + \left( \frac{v_{l_{\text{lat}}}}{v_{\text{max}}} \right)^2 - 1 & \text{if } v_f > 0 \\
\left( \frac{v_f}{v_{\text{min}}} \right)^2 + \left( \frac{v_{l_{\text{lat}}}}{v_{\text{max}}} \right)^2 - 1 & \text{if } v_f < 0 
\end{cases}$$

(1)

where $v_f$ and $v_{l_{\text{lat}}}$ are respectively the frontal and lateral speed, and $v_{\text{max}}$, $v_{\text{min}}$ and $v_{l_{\text{max}}}$ their minimum and maximum values (See Figure 4). $D(q_{n_{\text{state}}}^n, q_{n+1_{\text{state}}})$ can be then computed by integrating this speed constraint along it.

Having expressed the successor operator, which allows the optimizer to choose which node to expand at each iteration, the A* evaluation function can be defined.

The Evaluation Function $\hat{f}(q_{n_{\text{state}}}^n)$: It is the estimated cost of an optimal path going through $q_{n_{\text{state}}}^n$ from $q_s$ to $q_f$ and can be written as:

$$\hat{f}(q_{n_{\text{state}}}^n) = \hat{g}(q_{n_{\text{state}}}^n) + \hat{h}(q_{n_{\text{state}}}^n)$$

(2)

where $\hat{g}(q_{n_{\text{state}}}^n)$ is the estimated cost of the optimal path from $q_s$ to $q_{n_{\text{state}}}^n$ and $\hat{h}(q_{n_{\text{state}}}^n)$ is a heuristic function giving the estimated cost of the optimal path from $q_{n_{\text{state}}}^n$ to $q_f$.

$\hat{h}(q_{n_{\text{state}}}^n)$ must verify $\hat{h}(q_{n_{\text{state}}}^n) \leq h(q_{n_{\text{state}}}^n)$ to ensure that the algorithm is admissible, i.e. the path from $q_s$ to $q_f$ is optimal. Since the robot is fastest while walking straight forward in the absence of obstacles, $\hat{h}(q_{n_{\text{state}}}^n)$ is defined as:

$$\hat{h}(q_{n_{\text{state}}}^n) = D(q_{n_{\text{state}}}^n, q_{n+1_{\text{front}}}) + \sum_{k=1}^{N_{\text{sample}}-n-2} D(q_{n_{\text{front}}^k}, q_{n_{\text{front}}^{k+1}}) + D(q_{n_{\text{front}}^{n+1}}, q_f)$$

(3)

where $N_{\text{sample}}$ is the total number of initial sample configurations in $P$ including $q_s$ and $q_f$. $\hat{h}(q_{n_{\text{state}}}^n)$ thus sums the cost of walking along $P$ while staying tangential to the path with the start and end transition costs from $q_{n_{\text{state}}}^n$ to $q_f$.

Now that the A* functions are fully defined, a search algorithm can be run to compute an optimal path $P_{\text{opt}}$ by changing the orientation of each sample node. An example is shown in Figure 5.

### 2.2 Motion Generation for a Humanoid Robot

A collision-free path $P$ for the 3-DoF bounding box can be found using RRT-Connect and RO. The regular sampling optimization (RSO), which is the subject of this paper, is then applied on the path and produces a path $P_{\text{opt}}$ that gives priority to nonholonomic motion.

Once the bounding box trajectory is computed, the robot has to walk along it. A footstep sequence is thus generated along $P_{\text{opt}}$ by geometric decomposition of the path, and the pattern generator cited in subsection 1.1 then produces the robot whole-body trajectory.

### 3 EXAMPLES

This section presents experimental results of the path optimizer after it has been inserted in the previously described walk planning scheme. Distance parameters $v_{\text{max}}$, $v_{\text{min}}$, $v_{l_{\text{max}}}$ are set to 0.5, 0.1, and 0.25 respectively. $d_{\text{sample}}$ is equal to $\frac{h}{6}$, where $h$ is the humanoid’s height. Tests are performed on a 2.13 GHz Intel Core 2 Duo PC with 2 GB RAM.

Simulations of the humanoid robot HRP-2 are run in three scenarios. The first one is a small en-
Table 1: Columns 1 to 5: Computational time (ms) of each planning stage for the presented scenarios. Columns 6 and 7: Humanoid robot walk time (s) for the presented scenarios using RO alone and a RO-RSO combination.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>RRT-Connect</th>
<th>RO</th>
<th>RSO</th>
<th>Robot Trajectory</th>
<th>Total</th>
<th>RO</th>
<th>RO+RSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chairs</td>
<td>3,968</td>
<td>1,887</td>
<td>2,144</td>
<td>66,140</td>
<td>74,140</td>
<td>40</td>
<td>35</td>
</tr>
<tr>
<td>Galton</td>
<td>91.69</td>
<td>2,497</td>
<td>237.8</td>
<td>65,730</td>
<td>68,560</td>
<td>66</td>
<td>57</td>
</tr>
<tr>
<td>Apartment</td>
<td>1,212</td>
<td>2,425</td>
<td>2,412</td>
<td>222,800</td>
<td>228,800</td>
<td>200</td>
<td>120</td>
</tr>
</tbody>
</table>

Figure 6: Perspective view of the simulated HRP-2 trajectory on the final optimized path passing between two chairs.

The planning scheme is finally applied in the apartment scenario. In Figure 8, it is evident that HRP-2 walks facing forward through the doors. As with previous scenarios, the trajectory is more realistic than a trajectory where RSO is not used. The added computation time for RSO is 2,412 ms, which is insignificant compared to the 228 s it takes for the whole planning scheme.

Additionally, since the environment is significantly larger and more constrained than the previous ones, the walk time difference is more striking: Table 1 shows that it takes the robot 80 s less to cross the apartment when an RO-RSO combination is used.

### 3.2 “Galton” Scenario

An uncluttered environment is considered in this case, and it can be seen that RRT-Connect and RSO computation times are very low compared to other environments. This can be explained by the fact that a tree connecting start and goal configurations is easier to find, and that the frontal orientation state is valid for all considered samples on the path (See Figure 7).

### 3.3 “Apartment” Scenario

The planning scheme is finally applied in the apartment scenario. In Figure 8, it is evident that HRP-2 walks facing forward through the doors. As with previous scenarios, the trajectory is more realistic than a trajectory where RSO is not used. The added computation time for RSO is 2,412 ms, which is insignificant compared to the 228 s it takes for the whole planning scheme.

Additionally, since the environment is significantly larger and more constrained than the previous ones, the walk time difference is more striking: Table 1 shows that it takes the robot 80 s less to cross the apartment when an RO-RSO combination is used.

Figure 7: Perspective view of HRP-2 optimized trajectory in the Galton board scenario.
4 CONCLUSION

In this paper, a novel simple optimization method is presented for humanoid walk planning that relies on a decoupling between trajectory and robot orientations. It uses an A* search that takes as input a path for the robot bounding box, and produces a path where a discrete set of configurations have been reoriented to generate a realistic time-optimal walk trajectory. Results show that new trajectories are more satisfactory while the added computation time is insignificant compared to the whole planning time.

Of course, this approach can be used in other fields such as graphical animation for digital actors to adapt the body orientation with respect to the goal during locomotion. With a motion capture library containing pre-recorded nonholonomic and holonomic walk behaviors, it is possible to lay this behavior on the actor trajectory and produce realistic movements.

ACKNOWLEDGEMENTS

This work was supported by the French FUI Project ROMEO.

REFERENCES


