INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Advanced Robotic Systems

State Generation Method for Humanoid Motion Planning Based on Genetic Algorithm

Regular Paper

Xuyang Wang, Tiansheng Lu and Peiyan Zhang

School of Mechanical Engineering, Shanghai Jiaotong University, China
* Corresponding author E-mail: xuyang_wang@msn.com

Originally published in the Journal of Humanoids, ISSN 1996-7209

DOI: 10.5772/50918

Abstract A new approach to generate the original motion data for humanoid motion planning is presented in this paper. And a state generator is developed based on the genetic algorithm, which enables users to generate various motion states without using any reference motion data. By specifying various types of constraints such as configuration constraints and contact constraints, the state generator can generate stable states that satisfy the constraint conditions for humanoid robots.To deal with the multiple constraints and inverse kinematics, the state generation is finally simplified as a problem of optimizing and searching. In our method, we introduce a convenient mathematic representation for the constraints involved in the state generator, and solve the optimization problem with the genetic algorithm to acquire a desired state. To demonstrate the effectiveness and advantage of the method, a number of motion states are generated according to the requirements of the motion.

Keywords state generation, multiple constraints, inverse kinematics, genetic algorithm, humanoid robot

1. Introduction

From the viewpoint of motion planning, the continuous motion of a humanoid robot can be represented by a series of key states, and each state corresponds to a specified body configuration during the motion (Wang 2007 et al.). To define these key states, numerous and effective original motion data are essential. Currently, the original motion data are acquired primarily through the method of HMCD, i.e. Human Motion Capture Data. But these motion data cannot be applied directly to the humanoid robot because of kinematics and dynamics inconsistence between the human subject (whose motion is recorded) and the humanoid. Techniques such as motion warping (Witkin et al. 1995) or dynamic filtering (Yamane et al. 2000, Tak et al. 2005) are often used to ensure that the captured motions can be transformed into a dynamically feasible one. However, these methods are not flexible as dealing with the contact constraints and stability maintenance. An effective state should satisfy various kinds of constraints, such as configuration constraints, contact constraints, stability constraints, and so forth. So, it is a problem of multiple constraints. At the same time, the coordinate transformations between the task and actuator space are necessary in the course of motion planning, which certainly relates to the solving of inverse kinematics. As redundant manipulators, humanoid robots are still under development in the areas of multiple constraints and inverse kinematics.

In most of the existing biped systems, the synergy method (Vukobratovic et al. 1990, Amos 2003) and methods based on optimization (Capi et al. 2002) are often used. But for an irregular motion, e.g. standing up from a lying to a standing posture, the computation for ideal trajectories becomes impractical because of high computational efforts. So, the methods lack applicability for the design of various complex motions. Yamane et al. proposed a computational technique for creating whole-body motions of human without any captured motion (Yamane 2003). By specifying arbitrary constraints to the links and joints, the animators can generate a natural motion accordingly. The method also presents an intuitive pin-and-drag interface where the user can generate whole-body motion by simply switching on or off or strengthening or weakening the constraints. The pin-and-drag method shows good performance for multiple constraints and inverse kinematics with redundancy. But it doesn't take into account the stability of the robot during the motion. On the other hand, it is unavoidable to compute Jacobian matrices frequently for introducing the differential kinematics to solve inverse kinematics. Hauser et al. (2005) studied non-gaited humanoid locomotion planning and proposed a "contact-before-motion" approach. The planner allows contact with any pre-designated part of the robot's body and uses a probabilistic, sample-based approach to compute each step. In this approach, all possible contacts (that is a point from the environment and a point of the robot) are given at the beginning, and a method of iterative constraint enforcement is adopted to accelerate the sampling of all feasible configurations. Based on the similar idea, Escande et al. (2006) developed "contact-before motion" approach by incrementally building a contact tree according to a potential-like function. Starting with a description of the environment and of a target, the planner can compute a sequence of postures that allow the robot to reach its target. Their works develop the studies for whole-body contactable motions by considering the multiple constraints in the planner.

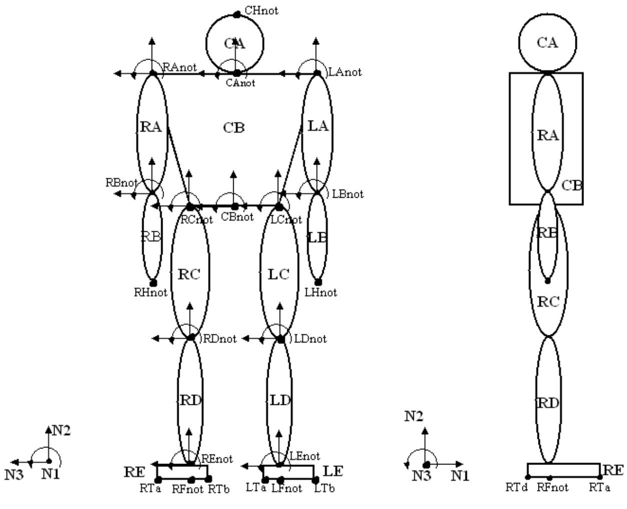
This paper describes a new generation method of the original motion data based on the genetic algorithms (the GAs). Since the algorithm generates key states for humanoid motion planning, it is also termed a State Generator in our study. The State Generator provides a new motion data generation method for the whole-body motion planning for the humanoid robot. Without utilizing any reference motion data, the State Generator can create various desired motion states by specifying arbitrary configuration and contact constraints.

The remainder of this paper is organized as follows: we first introduce a kinematics modeling method for humanoid robot in Section 2 and then present the mathematics representation for various constraints in Section 3. Section 4 describes the state generation process based on the genetic algorithms. In Section 5, some motion examples are shown to validate the proposed method. Finally, we summarize the contributions of this paper in Section 6.

2. Kinematics Modeling of the Humanoid Robot

Kinematics modeling is to establish the relationship of the joints and links of the robot, which is a fundamental task for the motion planning. This work consists of two parts: the forward kinematics and the inverse kinematics. Generally, the inverse kinematics solution is much more difficult than the forward one. In the motion planning and control of the robot, in order to reach a desired location for the end-effector of the manipulator, suitable values of the joint variables must be specified. The humanoid robot is a typical redundant robot, so mapping from the world coordinates to the joint coordinates is not one to one, meaning that there may exist an infinite number of joint variables setting which result in a given end-effector position/orientation. At present, minimizing an error function through some iterative optimization technique has been the main approach for solving the inverse kinematics problem of redundant robots. This function corresponds to the Euclidean distance between the actual and the target position/orientation of the robot's end-effector. But for a humanoid robot, more complex factors have to be taken into account in the real-life application, such as the contact constraints, the stability maintenance, and so on. In this case, the error function must be modified to combine the desired optimization criteria. It is very disadvantage for the solution of the problem.

In this paper, we select the GAs as the optimization algorithm for solving the inverse kinematics problem for the humanoid robot. The GAs solution needs only the forward kinematics equations which are easily to acquire, and the cost function (fitness function) which envelops various constraints has not the requirements of continuity in the derivatives. The characteristics of the GAs will be presented particularly in the Section 4. In this Section, we primarily introduce the modeling method for the forward kinematics.



(a) (b)

Figure 1. Structure model of the humanoid robot in this study. Image (a) is a front view and (b) is a side view

An outline of the model structure developed in our study is shown in Fig. 1. The model has 12 rigid body segments in total, and each rigid body is represented by two capitalized characters, for example CB, CA, etc. The total degree of freedom (DOF) of the humanoid robot is 17. In details, the DOF is distributed as: each leg has 5 DOF (2 for the hip joint, 1 for the knee joint, and 2 for the ankle joint), each arm has 3 DOF (2 for the shoulder joint, 1 for the elbow joint), and 1 DOF for the head. We represent the robot as a mechanism of tree structure whose root is a free-flying base link (pelvis) i.e. CB of the model. Taking a point on the root link CB, i.e. CBnot, as the reference point, we can establish the kinematics equations gradually. Thus, to describe the movement of the whole-body of the robot in the global coordinates N (where N1, N2, N3 are three unit base vectors), 23 generalized coordinates are needed totally: 6 of them are used to represent the configuration of the pelvis in the global coordinates, and the rest 17 of them are used to describe the relationship of the other parts of the robot relative to the pelvis. In this way, the movement of any point $P_i(i=1,2,\cdots v)$ to be studied on the robot (where these specified points are termed character points, such as the points CHnot, RHnot, RDnot, and so on) can be represented by

$$x_i^1 = \mathbf{r}_i \bullet \mathbf{N}1 \quad x_i^2 = \mathbf{r}_i \bullet \mathbf{N}2 \quad x_i^3 = \mathbf{r}_i \bullet \mathbf{N}3 \qquad i=1,2,\cdots v \tag{1}$$

Here, $\mathbf{r}_i$ is the position vector of the characteristic point P_i in N; x_i^k is the coordinate of the point P_i with regard to the coordinate axe $\mathbf{N}_k\,(k=1,2,3)$. Consequently, we can get the fallowing kinematics equations that describe the movements of the character points:

$$\mathbf{x} = \mathrm{f}\left(q_1, q_1, \cdots q_n\right) \tag{2}$$

Where: $\mathbf{x} = \left(x_1^1, x_1^2, x_1^3, \cdots, x_v^1, x_v^2, x_v^3\right)^T$, q is the generalized coordinates, and the number of the generalized coordinates n is equal to 23.

3. Definition and Formulation of the Constraints

3.1 Configuration Constraints

The constraints of the kinematic systems can be classified into two categories, that is, holonomic and nonholonomic constraints. Since the State Generator is primarily to generate desired robot states that satisfy some configuration requirements, it belongs to the category of holonomic constraints. The space configuration of the robot can be characterized by specifying whole or partial character point coordinates, and partial joint angles. Suppose N_P character points which positions can be expressed as $\mathbf{x}_{Pi}(i=1,\cdots,N_P)$, and N_Q joints which angles are $\theta_{Qi}=(i=1,\cdots,N_Q)$, are designated as the configuration constraints. Then all of these constraints can be expressed as vector as

$$\boldsymbol{\Gamma} = \left(\mathbf{x}_{P1}, \cdots, \mathbf{x}_{PN_P}, \theta_{Q1}, \cdots, \theta_{QN_Q}\right)^T \tag{3}$$

In general, robot's states alternate with the constrained subjects, therefore the members in $\boldsymbol{\Gamma}$ are changed frequently in accordance with the generation of different states. In order to represent the constraints in a uniform expression, we introduce a selection matrix $\mathbf{M}$ which consists of elements 0 or 1 to pick up the constrained variables, including all of the character point coordinates denoted as $\mathbf{x}_P$, and all of the joint angles denoted as $\boldsymbol{\theta}_Q$. Thus

$$\mathbf{M} = \begin{pmatrix} \delta_1 & & 0 \\ & \ddots & \\ 0 & & \delta_m \end{pmatrix} \tag{4}$$

where $\delta_i \in \{0,1\}$, moreover $\delta_i = 1$ if the corresponding variable is to be constrained, and $\delta_i = 0$ if not; m is the total number of the possible constraint variables. Especially, $\mathbf{M}$ can be expressed as

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_P & 0 \\ 0 & \mathbf{M}_Q \end{pmatrix} \tag{5}$$

here $\mathbf{M}_P$ and $\mathbf{M}_Q$ are the selection matrices which correspond to the character coordinates $\mathbf{x}_P$ and the joint angles $\boldsymbol{\theta}_Q$, respectively. Consequently, according to defining different selection matrix $\mathbf{M}$, the corresponding constraint vector $\boldsymbol{\Gamma}$ can be acquired.

3.2 Static Stability Constraints

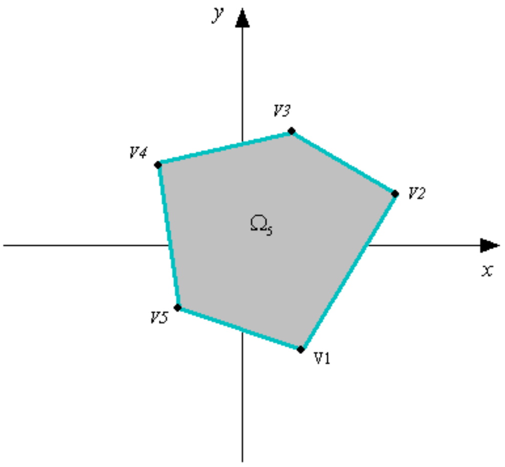


Figure 2. Vertex representation of a convex hull acquired by using Graham scan. The lowest point V_1 is the pivot, and the other points are distributed in a counter-clockwise manner

In order to guarantee the generated states are static stable, the static stability constraints must be taken into account, that is, the ground projection of the center of the gravity (COG) of the robot should be kept in the support polygon. Two aspects of the contents are involved in this study: one is the mathematic expression of the support polygon; and the other is the relativity calculation for the ground projection of the COG relative to the support polygon.

1) Mathematic Expression of the Support Polygon

Idealistically, an arbitrary point on the robot can be taken as the support point interacting with the environment. However, in the real-life circumstance, the contact points primarily focus on the ends and the joints of the limbs. In this paper, we utilize the predefined character points $P_i(i=1,2,\cdots v)$ as the possible contact points, and introduce a selection matrix $\mathbf{M}_C$ to specify the vector of the contact points in accordance with the robot's state. The definition of the contact selection matrix $\mathbf{M}_C$ is similar to the definition of $\mathbf{M}$, that is, $\delta_i=1$ if the character point p_i has the contact requirement, and $\delta_i=0$ if not. According to $\mathbf{M}_C$, we can find the contact points and then calculate theirs space coordinates. The fallowing question is how to form the support polygon in accordance with these discrete points. Apparently, it is a problem of convex hull. The convex hull of a set of points is the smallest convex set that includes the points. For a two dimensional finite set the convex hull is a convex polygon, which can be represented with a vertex representation method mathematically, i.e.

$$\left\{ \sum_{i=1}^{l} \lambda_i V_i \middle| \sum_{i=1}^{l} \lambda_i = 1 \text{ and } \lambda_i \geq 0 \ \ \forall i=1,\cdots,l \right\} \tag{6}$$

There are a lot of algorithms for computing the convex hull, and Graham scan is the most popular one (O'Rourke 2005). Given a set of points on the plane, Graham scan works in three phases to compute the convex hull:

a) First, select a point as the pivot. This point is guaranteed to be on the hull, and is chosen to be the point with minimum y coordinate. We select rightmost point in the set in case of the tie.
b) Next, sort the points in order of increasing angle about the pivot. We end up with a star-shaped polygon.
c) Build the hull, by marching around the star-shaped polygon, adding edges when we make a left turn, and back-tracking when we make a right turn.

For the detailed algorithm, please refer to O'Rourke (2005). As shown in Fig. 2, we acquire a convex hull Ω_5 by utilizing Graham's scan. And the polygon Ω_5 consists of vertexes $V_1,V_2,\cdots,V_5$, where V_1 is the lowest vertex and $V_1,V_2,\cdots,V_5$ distribute in counter-clockwise.

2) Position Determination of the COG

The position vector of the COG of the robot can be expressed as

$$\mathbf{r}_G = \frac{\sum_{i=1}^{12} m_i \mathbf{r}_{G_i}}{\sum_{i=1}^{12} m_i} \tag{7}$$

where $\mathbf{r}_G$ is the position vector of the COG of the robot; $\mathbf{r}_{G_i}$ is the position vector of the particle G_i; m_i is the mass of G_i. To calculate whether the ground projection of the COG, i.e. G', lies inside the support polygon, we can firstly choose a definite interior point of the convex polygon such as the geometrical center point C, then check whether G' locates at the same side with C for all convex edges. If G' and C locate at the same side for all convex edges, G' lies inside the convex polygon, otherwise G' lies outside. We can use the cross product to calculate whether two points locate at the same side of a line. For instance convex polygon Ω_3 shown in Fig.3, the cross product

$$\mathbf{a}_i = (V_{i+1} - V_i) \times (C - V_i) \tag{8}$$

$$\mathbf{b}_i = (V_{i+1} - V_i) \times (G' - V_i) \tag{9}$$

For all $i=1,2,3$, define $V_4 = V_1$, and if the z components of the cross products $\mathbf{a}_i$ and $\mathbf{b}_i$ have the same signs (i.e., their product is positive), then G' lies inside the convex polygon Ω_3, otherwise G' lies outside of Ω_3. Similarly, the method is still applicable for all $\Omega_l(l \geq 3)$, as long as defining $i=1,2,\cdots,l$ and $V_{l+1} = V_1$.

4. State Generation Method Based on GAs

Genetic algorithms (GAs) are search procedures based on the mechanics of natural genetics and natural selection. They combine an artificial survival of the fittest with genetic operators abstracted from nature to form a surprisingly robust search mechanism that is suitable for a variety of search problems (Goldberg 1989). There are several attractive features of GAs that make them suitable to solve the inverse kinematics problem of the humanoid robots (Parker 1989, Andreas 1998). The cost function is not necessarily required continuous in the derivation, so arbitrary form of the cost function can be selected for extremizing. There is no need for GAs to calculate the Jacobian Matrix of the end-effector relative to the joint variables. The GAs solution need only the forward kinematics equations, that is usually easy to acquire. The joint rotation limits can be handled directly, so any solution determined by GAs is physically realizable.

The main task of the State Generator is to generate the robot states that are compatible with the configuration and stability constraints by utilizing the effective searching technique of the GAs. Here the issue to be dealt with for the State Generator is not a single inverse kinematics problem, but a hybrid optimization problem that combine inverse kinematics with stability problems.

As shown in the followings, the GA is basically composed of five components that affect the algorithm significantly.

1) Encoding of the variables
2) Initial population's setting
3) Design of the fitness function
4) Genetic operation rules
5) Control of the algorithm parameters

One advantage of the GAs is its no requirement for the special knowledge about the optimization, where the optimal solution can be found by evaluating the fitness values of the chromosome-like data according to the defined fitness function. Suppose the optimization variables are $\boldsymbol{\varphi}=\left(q_1,q_2,\cdots,q_n\right)^T$, here $n=23$. It means the optimization variables are equivalent to the generalized coordinates actually, so we formulate all the constraint conditions as:

1) The boundary condition of the optimization variables: $\mathbf{LB}\le\boldsymbol{\varphi}\le\mathbf{UB}$
2) The constraints of the joint angles: $\mathbf{M}_Q(\boldsymbol{\varphi}_Q-\tilde{\boldsymbol{\theta}}_Q)=0$
3) The position constraints of the character points: $\mathbf{M}_P(\mathbf{x}-\tilde{\mathbf{x}}_P)=0$
4) The position constraints of the contact points: $\mathbf{M}_C(\mathbf{x}-\tilde{\mathbf{x}}_C)=0$

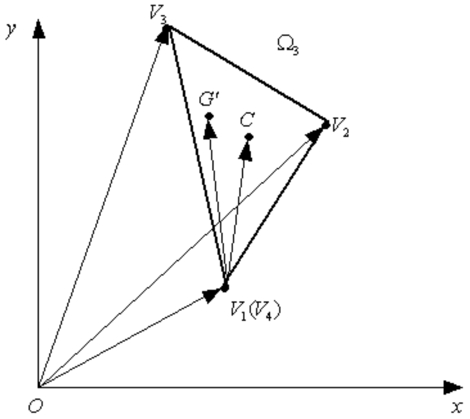


Figure 3. Checking the relative position of G' (the ground projection of the COG) with respect to the support polygon Ω_3

Static stability constraints: for all $i=1,2,\cdots,l$, $\mathbf{a}_i \bullet \mathbf{b}_i > 0$

Here, $\mathbf{UB}$ and $\mathbf{LB}$ are the upper and lower boundary conditions, respectively; $\boldsymbol{\varphi}_Q=\left(q_4,q_5,\cdots,q_n\right)^T$ is the subset of $\boldsymbol{\varphi}$ and corresponds to the joint variables including the orientation of the trunk in the global coordinates; $\tilde{\boldsymbol{\theta}}_Q$ ($\tilde{\mathbf{x}}_P$ and $\tilde{\mathbf{x}}_C$) are the corresponding values of the constraint variables $\boldsymbol{\theta}_Q$ ($\mathbf{x}_P$ and $\mathbf{x}_C$).

In these constraint conditions, constraint 1 and 2 can be met naturally by encoding of the variables; whereas constraint 3, 4 and 5 have to be synthetically satisfied according to specifying a proper fitness function. In this paper, the fitness function is designed as

$$\mathrm{F}(\boldsymbol{\varphi},k)=\left\|\mathbf{M}_P(\mathbf{x}-\tilde{\mathbf{x}}_P)\right\|+\left\|\mathbf{M}_C(\mathbf{x}-\tilde{\mathbf{x}}_C)\right\|+k\,\mathrm{P}(\mathbf{a},\mathbf{b}) \quad (10)$$

Where $\|\cdot\|$ denotes the Euclidean distance; k is the penalty factor; $\mathbf{P}(\mathbf{a},\mathbf{b})$ is the penalty term of the stability constraints, which can be expressed as

$$\mathrm{P}(\mathbf{a},\mathbf{b})=c_1\sum_{i=1}^{l}\{\min[0,\mathbf{a}_i\bullet\mathbf{b}_i]\}^2+c_2\left\|G'-C\right\| \qquad (11)$$

here, c_1, c_2 are the unbalanced weight numbers that give rise to the penalties when the ground projection G' leaves the support area Ω_l i.e. $\min[0,\mathbf{a}_i\bullet\mathbf{b}_i]\neq 0$, and G' deviates from the geometrical center C of the support area i.e. $\left\|G'-C\right\|\neq 0$, respectively. The values of c_1 and c_2 can be determined according to the process of testing, in general, $c_1 \gg c_2$ is preferential.

5. Examples

The State Generator is implemented based on the software of Matlab by utilizing its Genetic Algorithm Toolbox, where the proposed method is employed to generate the key states for the whole-body motion planning for the humanoid robot. In the generalized coordinates, i.e. the optimization variables, $q_{1,2,3}$ and $q_{4,5,6}$ are the position coordinates and orientation angles of the trunk (CB) in the global coordinates **N** (**N**1, **N**2, **N**3), respectively. And the others, also known as the joint variables are defined as Table 1.

Joints	DOF	Segments	**W** **(Origin)**	**W1**	**W2**	**W3**
neck	1	CB CA	**CB** (CBnot)	-	q_7	-
right shoulder	2	CB RA	**CB** (CBnot)	q_9	-	q_8
right elbow	1	RA RB	**RA** (RAnot)	-	-	q_{10}
left shoulder	2	CB LA	**CB** (CBnot)	q_{12}	-	q_{11}
left elbow	1	LA LB	**LA** (LAnot)	-	-	q_{13}
right whist	2	CB RC	**CB** (CBnot)	q_{15}	-	q_{14}
right knee	1	RC RD	**RC** (RCnot)	-	-	q_{16}
right ankle	2	RD RE	**RD** (RDnot)	q_{18}	-	q_{17}
left whist	2	CB LC	**CB** (CBnot)	q_{20}	-	q_{19}
left knee	1	LC LD	**LC** (LCnot)	-	-	q_{21}
left ankle	2	LD LE	**LD** (LDnot)	q_{23}	-	q_{22}

Table 1. Definition of the joint variables. That is, the joints and their degree of freedom implemented in the model. Each joint is formed by two adjacent segments that can be found in Fig 1. And the reference coordinates represented by **W** (**W**1, **W**2, **W**3) with origins XXnot are used to formulate the relative configuration of the segments

5.1 Regulation of the Ground Projection of the COG

The original state generated by the State Generator is shown in Fig. 4, that is, an erect standing posture of the humanoid robot in the simulation space.

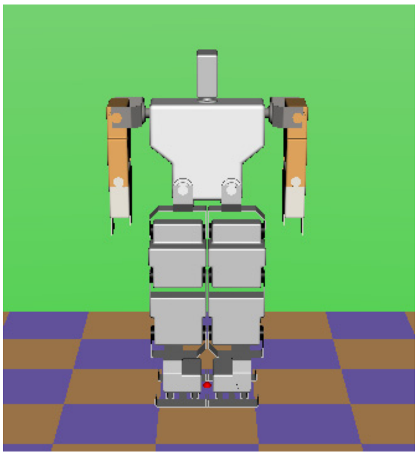
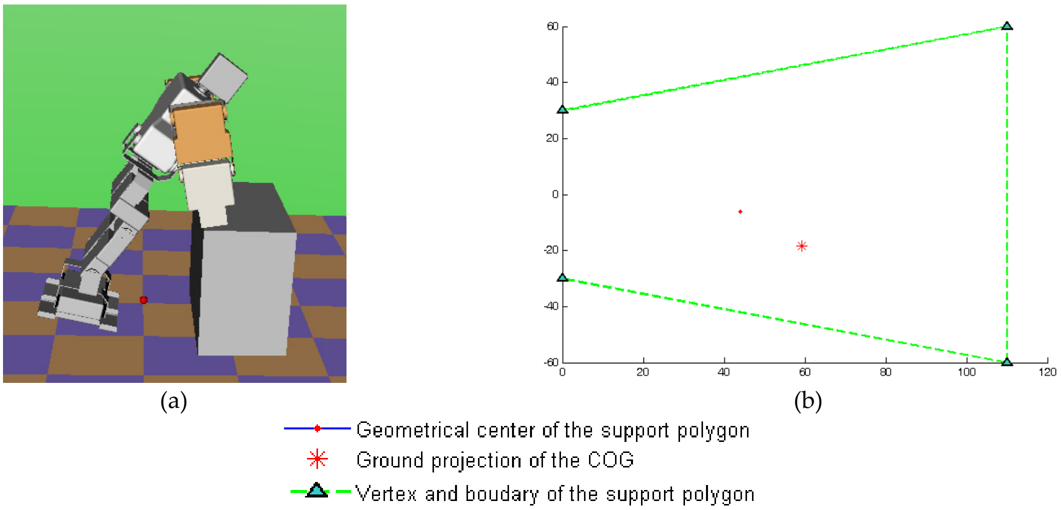


Figure 4. Initial output of the State Generator, i.e., the original state of the simulator without applying any constraints

In Fig. 5, we give an instance to illustrate the graphic ouputs of the State Generator, where the robot is standing with both hands touching an object. From Fig. 5(b), we can see that the ground projection of the COG locates inside the support polygon, which indicates the generated state is static stable. However, it's obvious that the ground projection of the COG is not strictly coincident with the center of the support polygon. In this case, we can increase the unbalanced weight number c_2 properly to acquire more stable states if it is required. Fig. 6 illustrates the results after increase c_2 from 500 to 800, where the ground projection of the COG is moved near to the geometrical center of the support polygon consequently. This becomes very convenient when we deal with some states which are very sensitive to the stability.



(a)

(b)

Figure 5. Graphic representations for the outputs: picture (a) is the generated state of the robot whose hands touching an object, and graph (b) is the distribution of the support polygon vertexes as well as the ground projection of the COG

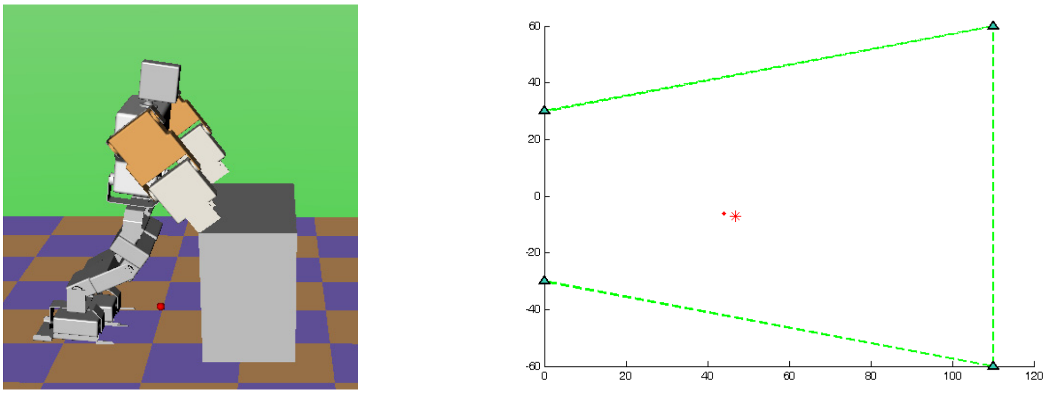


Figure 6. The state generated by increasing the unbalanced weight number to acquire more stable one

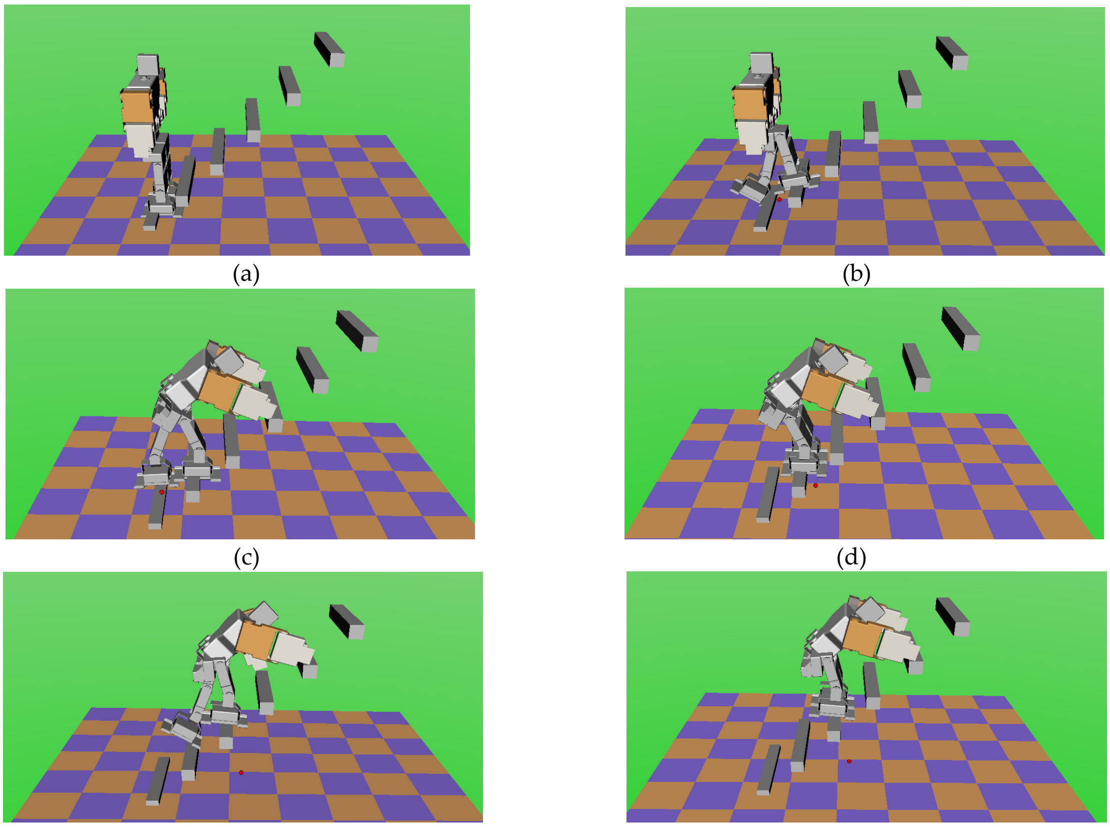

(a)
(b)
(c)
(d)
(e)
(f)

Figure 7. Climbing a ladder. (a) The initial state before climbing. (b) The robot begins to climb. (c) The robot has contacts on both feet and hands. (d) The right foot moves up one step. (e) The contacts are changed for the next climbing cycle. (f) The robot is then moved up one step again

5.2 State Generations

Fig. 7 shows a series of key states generated by the State Generator to demonstrate a ladder climbing behavior. In this example, the hands are controlled to be in contact in order to maintan balance and also to control the ascention of the center of the mass. The climbing cycle is produced by designing the foot and hand positions at the ladder contacts. This example demonstrates the usefulness of the developed algorithms.

In this section, we summarize this paper as followings:

1) We present a particular mathematic representation for the constraints involved in the state generation, and propose a selection matrix to represent various constraints in a uniform expression.
2) In the static stability, we propose a convenient evaluation method by using Graham Scan to compute the convex hull for the support polygon.
3) With Genetic Algorithm, we simplify the state generation problem that involves the multiple constraints and inverse kinematics as a problem of

optimizing and searching, which provides a new idea for the generation of original motion data.

4) This study contributes to not only the original motion generation for humanoid motion planning, but also the similar problems that relate to the multiply constraints and inverse kinematics.

6. References

[1] Amos, A., Wilfried, G..: Analytic path planning algorithms for bipedal robots without a trunk. *J. Intell. Robot. Syst*. 36(2), 109 - 127(2003 February)

[2] Andreas C.N.: Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm. *Mechanism and Machine Theory*. 33(3), 273–292 (1998, April)

[3] Capi G., Kaneko, S., Mitobe, K., Barolli, L., Nasu, Y.: Optimal trajectory generation for a prismatic joint biped robot using genetic algorithms. *Robotics and Autonomous Systems*, 38(2), 119-128 (2002)

[4] Escande, A., Kheddar, A., Miossec, S.: Planning support contact-points for humanoid robots and experiments on HRP-2. In: *Int. Conf. on Intell. Robots Syst*., pp. 2974 - 2979. Beijing, China (2006, October)

[5] Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. *Reading*, MA: Addison-Wesley (1989)

[6] Hauser, K., Bretl, T., LatombeNon, J.C.: Non-gaited humanoid locomotion planning. In: *5th IEEE-RAS Int. Conf. Humanoid Robots,* pp. 7–12. Tsukuba, Japan (2005, December)

[7] O'Rourke, Joseph.: Computational geometry in C. China Machine Press, Beijing (2005)

[8] Parker, J., Khoogar, A., Goldberg, D.: Inverse kinematics of redundant robots using genetic algorithms. In: *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 271–276, 1989, Scottsdale USA.

[9] Tak, S., Ko, H.S.: A physically based motion retargeting filter. *ACM Trans. Graphics*, 24(1). 98-117 (2005)

[10] Vukobratovic, M., Borovac, B., Surla, D., Stokic, D.: Biped locomation. *Dynamics, stability, control and application*. Springer, Berlin Heidelberg New York (1990)

[11] Wang, X.-Y., Lu, T.-S., Zhang, P.-Y.: Study on state transition method applied to motion planning for a humanoid robot. *International Journal of Advanced Robotic Systems.* 4(3), 349–354 (2007, September)

[12] Witkin, A., Papovic, Z.: Motion Warping. In Cook, R. (ed.) Computer Graphics (*SIG GRAPH'95 Proceedings*). 105–108 (1995)

[13] Yamane, K., Nakamura, Y.: Dynamics filter - concept and implementation of online motion generator for human figures. In: *IEEE Int. Conf. Robot. Autom*., pp. 688–694. San Francisco, CA. (2000, April)

[14] Yamane, K., Nakamura, Y.: Natural Motion Animation through Constraining and Deconstraining at Will. *IEEE Transactions on Visualization and Computer Graphics*, 9(3), 352–360 (2003)