2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Getting up Motion Planning using Mahalanobis Distance

Fumio Kanehiro, Kiyoshi Fujiwara, Hirohisa Hirukawa, Shin'ichiro Nakaoka and Mitsuharu Morisawa

Abstract— **This paper presents a motion planner of getting up motion using Mahalanobis distance. It is an indispensable function for humanoid robots to get up by itself and some humanoid robots are able to get up by themselves, but the motion can start only from the states specified a prior. The robots have to get up from an arbitrary lying state which may result after an unexpected falling. The proposed method (1)determines the degree of similarity between the current falling state and predefined falling states using Mahalanobis distance, (2)generates a collision-free motion to the most similar state, and (3)plans a sequence of motions using a state transition graph.**

I. Introduction

Recent progress in the control of biped locomotion has realized stable biped locomotion on several humanoid robots including ASIMO[1], QRIO[2] and HRP-2[3]. No matter how the control is improved, a humanoid robot must have an ability to get up by itself after it has fallen on a floor. Because even a human sometimes falls over.

A trial to let the robot get up started for a simple robot with a few links. Morimoto et al. realized the acquisition of stand-up behavior of a robot with three links by hierarchical reinforcement learning, and the proposed method was examined by a simulation and experiment[4]. In recent years, the motions have been investigated for humanoid robots with many links. We have realized a getting up behavior on a small size humanoid robot which was 30[cm] height [5]. morph[6], HOAP2[7], QRIO and its prototype SDR-3X[8] are also small size humanoid robots that are able to get up. On the other hand, there are a few life size humanoid robots which can get up. HRP-2 can get up from lying states with the face upward and downward. R.Daneel Study 1[9] can get up quickly by swinging down its legs from a lying state on the back[10].

These getting up motions can start only from predefined falling states. But the robot may be in an arbitrary state when it falls unexpectedly. In order to get up by itself and continue working in the cases, the robot must be able to get up from any falling state.

This paper aims to realize the getting up ability which may start from an arbitrary lying state on the flat floor and the proposed method (1)determines the degree of similarity between the current falling state and predefined falling states using Mahalanobis distance, (2)generates a collision-free motion to the most similar state, and (3)plans a sequence of motions using a state transition graph.

The authors are with Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology(AIST), Tsukuba Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki, 305-8568 Japan {f-kanehiro, k-fujiwara, hiro.hirukawa, s.nakaoka, m.morisawa}@aist.go.jp

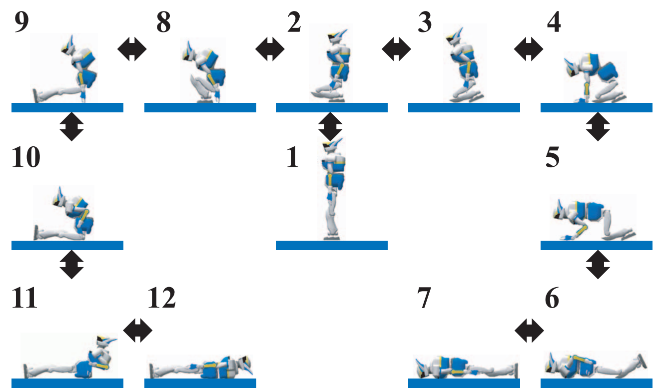

Fig. 1. State Graph for Getting up Motion

This paper is organized as follows. Section 2 overviews the proposed method, and Section 3 shows how to improve it. Section 4 concludes the paper with future works.

II. Getting up Motion Planning

A. Strategy

The getting up motions of HRP-2 was realized based on the state transition graph as shown in Fig.1[11]. The algorithm tries to find a path on the graph by a graph search algorithm and the transitions between the consecutive states are realized by the controllers assigned to the transitions.

When a robot falls over unexpectedly, the robot must get up from the lying state after the falling which may not be a state in the graph. If the robot can transfer from the state to any of the states in the graph, it should be able to get up by applying the available method to get up. The overall structure of this strategy is similar to that of PRM[12]. The graph is given not by learning but by hand and we devote our attention to the problem how to let the robot transfer from a given lying state to a state in the graph. The problem includes issues, (1)how to select a target state of the graph to transfer and (2)how to generate a motion to reach the state.

The proposed method tries to select a state which is most similar to the current falling state in the sense as defined below, and applies a linear interpolation to the posture of the robot to reach a state of the graph.

B. Target Selection using Maharanobis Distance

Let $\boldsymbol{q} = [q_1 \cdots q_n]^T$($n$ is the DOF) be the joint angle vector of a robot and $[\phi\ \theta\ \psi]^T$ Euler angles of the base link.

Since a lying state on the flat floor does not depend on ψ, a lying state $\boldsymbol{X}$ can be defined by

$$\boldsymbol{X} = [\phi\ \theta\ \boldsymbol{q}]^T.$$

Let d_j be the distance between the current lying state $\boldsymbol{X}$ and a state $\boldsymbol{X}_j (j = 1, \cdots, N)$ in the graph(N is the number of known states in the graph). $\boldsymbol{X}_j$ that has the minimum d_j is the most similar state to $\boldsymbol{X}$.

A simplest method to calculate d_j must be Euclidean distance which can be given by

$$d_j^2 = (\boldsymbol{X} - \boldsymbol{X}_j)^T (\boldsymbol{X} - \boldsymbol{X}_j).$$

But this method is not appropriate, since all joints and Euler angles are treated in the same way.

Another method is weighting each element in which d_j is given by

$$d_j^2 = (\boldsymbol{X} - \boldsymbol{X}_j)^T diag(\boldsymbol{w}) (\boldsymbol{X} - \boldsymbol{X}_j),$$

$$\boldsymbol{w} = [w_1\ ...\ w_{n+2}]^T.$$

We have applied the method to realize the getting up motion of a small size humanoid robot[13], but the problem was that there is no reasonable criterion for deciding an appropriate $\boldsymbol{w}$ other than an ad-hoc one.

Consequently we use Mahalanobis distance for evaluation of the degree of similarity. Then d_j is calculated by

$$d_j^2 = (\boldsymbol{X} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{X} - \boldsymbol{\mu}_j),$$

where $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are the average vector and a variance/covariance matrix of S_j which is a set of lying states which can transfer to $\boldsymbol{X}_j$ respectively.

C. Acquisition of the Average Vector and Variance/Covariance Matrix

$\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are acquired by the following procedure.

1) Generate a set of various lying states S using random numbers.
2) Try a transition from $\boldsymbol{X}_i (\boldsymbol{X}_i \in S)$ to $\boldsymbol{X}_j$ on a simulator[14] and add it to S_j if the transition succeeds.
3) Calculate $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ from S_j.

S is created by the following procedure.

1.1) Set values of q_i within its movable range using random numbers.
1.2) Check an occurrence of self-collision and go back to 1.1) if it occurs. Humanoid robots has wide movable ranges and several links are connected sequentially. This structure causes self-collision frequently. In this step, about 80[%] of generated postures cause self-collision.
1.3) Set values of ϕ and θ in range$[-\pi, \pi)$ using random numbers.
1.4) Set the position of the base link to $[0\ 0\ 1]^T$.
1.5) Start a simulation and stop it when both of linear velocity and angular velocity of the base link become smaller than the threshold values for a period of time. Finally make $\boldsymbol{X}$ from the current state and add it to S.

TABLE I
RESULTS OF TRANSITION TRIALS(UNIT:[%])

State No.	success	failure	collision	overload
5	2.5	43.9	17.6	36.1
6	3.4	0.6	20.9	75.1
7	5.2	42.2	15.1	37.5
10	4.1	15.8	40.3	39.8
11	25.1	2.3	33.5	39.1
12	42.1	5.3	15.1	37.5

Transition trials are done as follows.

2.1) Start a simulation using $\boldsymbol{X}_i (\boldsymbol{X}_i \in S)$ as an initial state. A duration of a linear interpolation is decided to be proportional to the maximum absolute difference of joint angles of $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$.
2.2) Stop the simulation and go back to 2.1) when an overload on a joint or self-collision is detected. These may happen since a transition motion is generated by a simple linear interpolation of postures.
2.3) After detecting the robot motion has stopped by the method in 1.5), compare the current state and $\boldsymbol{X}_j$ and add $\boldsymbol{X}_i$ to S_j if they looks identical. Whether states are identical or not is decided by

$$\arccos((\boldsymbol{R}'_B \boldsymbol{R}_B^T \boldsymbol{v})^T \boldsymbol{v}) < \epsilon,$$

where $\boldsymbol{v}$ is a vertical unit vector, $\boldsymbol{R}_B$ is a rotation matrix of the base link calculated using $\boldsymbol{X}_j$ and $\boldsymbol{R}'_B$ is also a rotation matrix of the base link gotten from the simulation result.

Consequently, the results of the simulations are categorized into four types, *Success*, *Failure*(a motion is successful but an orientation is different), *Overload* and *Collision*.

D. Simulation

1) Calculation of the Average Vector and Variance/Covariance Matrix: In order to get $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$, 6,000 random falling states are generated at first and transitions to the graph are tried. Since trials on the simulator take long time(It takes a few minutes to execute one trial.), state #5, #6, #7, #10, #11 and #12 in Fig.1 are selected and only transitions to those states are tried. They are selected because it is considered that they are similar to various falling states. Results of these trials are shown in Table I.

About 40[%] of the trials to state #12 succeeds and another 40[%] fails with an overload. It is considered that the result is caused by the arrangement of movable ranges of the joints. They are set to move arms and legs forward(Figure 2 shows a posture where joint angles are set to the center of each movable range.). Therefore, an overload on a joint of arms and legs may often occur when the robot falls with the face downward and the trials succeed when it falls with the face upward.

Fig. 2. Posture(joint angles are set to the center of movable ranges)

TABLE II
RESULTS OF SELECTION AND TRANSITION TRIALS(UNIT:[%], ALL FALLING STATES)

No.	selected	success	failure	collision	overload
5	8.4	15.5	2.4	11.9	70.2
6	10.3	3.9	1.0	11.7	83.5
7	20.5	18.5	1.0	11.7	66.3
10	1.8	38.9	22.2	27.8	11.1
11	9.6	69.8	1.0	18.8	10.4
12	49.4	72.9	0.4	14.6	12.1

As a result of the same reason, 40[%] of trials to state #5 and #7 fails with an overload and another 40[%] fails with different orientation.

In case of the trials to state #6, 70[%] fails with an overload and the rate of *Failure* is almost zero. Since a posture of state #6 is pulling its head up, an overload on neck joints occurs even if it falls with the face upward. 53[%] of *Overload* was caused by the neck joints.

State #7 and #12 have different orientations but the same $\boldsymbol{q}$. So their simulation results are same. Therefore they have same *Overload* and *Collision* ratios. If the robot lies on the back or on the face at the end of the simulation, a *Success* ratio to state #7 must be the same with *Failure* ratio to state #12. But they are slightly different. This is because the robot lies on its side in a few cases.

2) Target Selection and Transition Trial: Target states for new 1,000 falling states are selected using $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ which are calculated in the previous section and the transitions are tried. Results are shown in Table II. Success ratios in Table II increased compared to them in Table I. This means that the proposed method works effectively.

But the total success ratio is only 48.9[%]. So the half of transitions fails. One of major reasons is that some of 1,000 falling states can't transfer to any of selected six states. To confirm it, the transitions to all six states are tried in the same way as Table I and the result shows only 56.4[%] of them can transfer to any of six states. Hence the success ratio can't over the value. Table III shows result when falling states which can transfer to any of the states in the graph are applied to the proposed method. In the case, the total success ratio is 86.8[%]. From this result, it is expected that a transition will succeed in a high rate if the falling state can transfer to any of the states in the graph and it is important for increasing the success ratio to decrease lying states which can't transfer to any state.

TABLE III
RESULTS OF SELECTION AND TRANSITION TRIALS(UNIT:[%], ONLY FOR FALLING STATES WHICH CAN TRANSIT TO KNOWN STATES)

No.	selected	success	failure	collision	overload
5	2.7	86.7	0.0	0.0	13.3
6	4.3	16.7	4.2	4.2	75.0
7	9.8	69.1	3.6	1.8	25.5
10	2.7	46.7	26.7	20.0	6.7
11	13.7	87.0	1.3	3.9	7.8
12	67.0	95.2	0.5	0.0	4.2

```
Make_Collision_Free_Motion(q_s, q_e)
 1  qlist ← (q_s);
 2  do
 3      cinfo ← Check_Collision(q_s, q_e);
 4      if cinfo then
 5          do
 6              q_new ← Make_Collision_Free_Posture(cinfo);
 7              cinfo ← Check_Collision(q_s, q_new);
 8          while cinfo
 9      else
10          q_new ← q_e;
11      qlist.append(q_new)
12      q_s ← q_new
13  while q_s ≠ q_e
14  Return qlist;
```

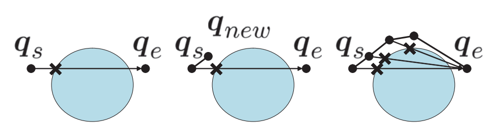

Fig. 3. Algorithm for making collision free motion

III. COLLISION-FREE TRANSITION MOTION

A. Collision-free Motion Generation

In order to decrease the lying states which can't transfer to any state in the graph, the occurrences of self-collision and an overload must be prevented. In this section, a collision-free motion generation method is introduced instead of a linear interpolation to remove failures caused by self-collision.

If an initial state and a final state are given in advance, a motion between them can be generated offline and its processing time is not restricted. However, in order to accept arbitrary lying states, the motion generation must be done online. So it is preferable that the generation finishes in several seconds. Therefore, the expensive motion planners can not be applied to this problem, and we used a method that may trapped in local minima but can generate a motion in a short period. It avoids self-collision locally using collision information.

Figure 3 shows an algorithm of the collision-free motion generation. It inputs an initial state $\boldsymbol{q}_s$ and a target state $\boldsymbol{q}_e$, inserts relay postures to avoid self-collision and outputs a sequence of postures $qlist$. $qlist$ is initialized using $\boldsymbol{q}_s$ in the first line. A transition to $\boldsymbol{q}_e$ is tried in the third line(Fig.3 left). Here *Check_Collision($\boldsymbol{q}_1, \boldsymbol{q}_2$)* is a function

```
Make_Collision_Free_Posture(cinfo)
 1  do
 2     q ← Posture(cinfo);
 3     δq ← 0;
 4     foreach pair in Collision_Pairs(cinfo)
 5        p ← Point(pair);
 6        n ← Normal(pair);
 7        L1, L2 ← Link(pair);
 8        J ← Jacobian(L1, L2, p);
 9        δq ← δq+ Expand(αJ⁺n);
10     q ← q + δq;
11     cinfo ← Check_Collision(q);
12  while cinfo
13  Return q;
```

Fig. 4. Algorithm for making collision free posture

which checks an occurrence of self-collision while transferring between $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ by a linear interpolation and returns collision information if self-collision is detected. In that case, a collision-free posture $\boldsymbol{q}_{new}$ is generated using collision information $cinfo$ in the sixth line(Fig.3 middle). In some cases, self-collision occurs in a transition between $\boldsymbol{q}_s$and $\boldsymbol{q}_{new}$. So it is checked in the seventh line and a $\boldsymbol{q}_{new}$ is generated again using a new $cinfo$ if self-collision is detected. This procedure is repeated until a collision-free transition to $\boldsymbol{q}_{new}$ is found and the found $\boldsymbol{q}_{new}$ is added to $qlist$(the 11th line) and it is assigned to $\boldsymbol{q}_s$(the 12th line). This procedure is repeated until a collision-free transition to $\boldsymbol{q}_e$ is found(Fig.3 right) and $qlist$ is returned at last.

B. Collision-Free Posture Generation

An algorithm of *Make_Collision_Free_Posture* in the sixth line of Fig.3 is shown in Fig.4. It inputs collision information $cinfo$ and outputs a collision-free posture $\boldsymbol{q}$. A collision detected posture is used as an initial value of $\boldsymbol{q}$ in the second line.

The following procedure is applied to each collision detected pair of links(the fourth line). A Jacobian matrix $\boldsymbol{J}$ of the joints between collision detected two links L_1, L_2 around a contact point $\boldsymbol{p}$ is calculated(the 8th line). A variation of joint angles is calculated using $\boldsymbol{J}$ and a contact normal $\boldsymbol{n}$ which has the equivalent length to the penetration depth(the 9th line). Here $\boldsymbol{J}^+$ is a pseudo inverse matrix of $\boldsymbol{J}$ and α is a positive value. A function *Expand* inserts zero and makes $\delta\boldsymbol{q}$ which has the DOF length. $\delta\boldsymbol{q}$ for all the collided pairs are added(the 9th line) and the final $\delta\boldsymbol{q}$ is added to $\boldsymbol{q}$(the 10th line). Then an occurrence of self-collision is checked(the 11th line) and this procedure is repeated while self-collision is detected.

A geometric model of a robot is represented by a set of triangles. And collision between them are tested by RAPID[15]. Since the collision check is done for discrete postures, many collisions occur at the same time. Let those contacts points $\boldsymbol{p}_i (i = 1, \cdots, N)$, normal vectors $\boldsymbol{n}_i$ and penetration depths d_i respectively. Here N is the total number of contacts. $\boldsymbol{p}$ and $\boldsymbol{n}$ are calculated by

$$\boldsymbol{p} = \frac{\sum_{i=1}^{N} d_i \boldsymbol{p_i}}{\sum_{i=1}^{N} d_i}, \ \boldsymbol{n} = \frac{\sum_{i=1}^{N} d_i \boldsymbol{n_i}}{N}.$$

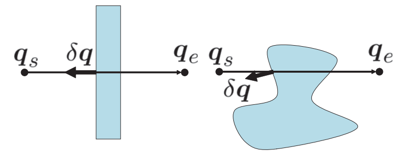

Fig. 5. Examples of situations which can't be solved by algorithm in Fig.4

```
9        δq ← δq+Expand(αJ⁺n)+Extract(β(q_h − q));
```

Fig. 6. Modified part of algorithm for making collision free posture

In many cases, it is possible to generate a collision-free posture using this procedure. But in some cases, a collision occurrence and a collision-free posture generation are repeated infinitely. Examples are shown in Fig.5. The left is the case where relative velocity of colliding links heads opposite direction to $\delta\boldsymbol{q}$. The right is the case where a motion goes into a pocket in the configuration space. These cases occur because only local collision information is used for collision avoidance.

To prevent occurrence of these situations, the 9th line of Fig.4 is modified as Fig.6. Here $\boldsymbol{q}_h$ is the posture in Fig.2 and β is a positive value. And function *Extract* extracts joint angles between L_1 and L_2. This third term adds an effect which pulls $\boldsymbol{q}$ to a collision-free posture in Fig.2 and then the cases in Fig.5 can be solved.

C. Motion Refinement

An acquired sequence of postures may include unnecessary postures since it is generated based on the local collision avoidance. So it is refined by an algorithm shown in Fig.7. This algorithm inputs a sequence of postures $qlist$ and outputs refined one $qlist_{new}$. $qlist_{new}$ and $\boldsymbol{q}_s$ are initialized using the head of $qlist$(the first and the second line). A posture to which $\boldsymbol{q}_s$ can transfer without self-collision is searched from the end of $qlist$(the 6th line). The found posture $\boldsymbol{q}$ is added to $qlist_{new}$ and is assigned to $\boldsymbol{q}_s$(the 8th and the 9th lines). This procedure is repeated until a collision-free transition to $\boldsymbol{q}_e$ is found.

The basic concept of the algorithm is shown in Fig.8. In case of Fig.8 left, $\boldsymbol{q}_0$ is used as $\boldsymbol{q}_s$ at first. The transitions are tried from $\boldsymbol{q}_5$ in turn and a successful transition from $\boldsymbol{q}_3$ is

```
Refine_Motion(qlist)
 1  q_s ← qlist[0];
 2  qlist_new ← (q_s);
 3  qlist_r ← Reverse(qlist);
 4  q_e ← qlist_r[0];
 5  while q_s ≠ q_e
 6     foreach q in qlist_r do
 7        if not Check_Collision(q_s, q) then
 8           qlist_new.append(q);
 9           q_s ← q;
10           Break;
11  Return qlist_new;
```

Fig. 7. Algorithm for removing unnecessary relay postures

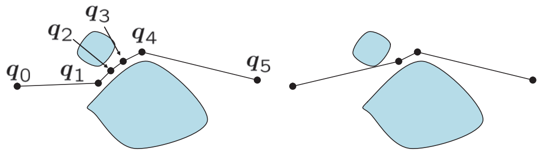

Fig. 8. Example of refinement of posture sequence(left:before, right:after)

Fig. 9. Example of collision free motion generation

found. Then $\boldsymbol{q}_3$ is set to $\boldsymbol{q}_s$. In the same way, $\boldsymbol{q}_4$ and $\boldsymbol{q}_5$ are added and a sequence of postures in Fig.8 right is acquired finally.

Figure 9 shows an actual example of this motion generation. A motion from a posture shown in Fig.9 leftmost to rightmost is generated. If the transition is done by a linear interpolation, the wrists of the robot collides. The algorithm Fig.6 is effective in the case since a relative velocity of the wrists and a contact normal goes into the opposite direction. A sequence of 77 postures is generated by *Make_Collision_Free_Motion* and it is reduced into five postures shown in Fig.9 by *Refine_Motion*. In this case, it takes 6.3[s] and 2.2[s] on Pentium4 3.4[GHz] to execute *Make_Collision_Free_Motion* and *Refine_Motion* respectively.

D. Simulation

The simulations in Section II-D are executed again using the collision-free transition motion generation method. Additionally, a part of judgment method is modified. If an overload on any of joints is detected, the simulation was stopped in the method of Section II-D. This time, even if an overload is detected on neck or finger joints which do not have strong structure and torque, the simulation is continued.

Table IV shows results of simulations of transitions from 6,000 falling states to six known states as Table I. As a judgment method is modified, an *Overload* ratio of the transition to state #6 decreased significantly. Instead a *Failure* ratio increased. Ratios of *Overload* and *Failure* became almost same as ratios of to state #5 and to state #7. As a transition motion is modified, *Success* ratios to state #11 and #12 increased. An *Overload* ratio to state #10 changed into *Failure* and *Collision* and a *Success* ratio did not change.

Figure 10 shows the averages of falling states which could transfer to state #6(left) and could not because of

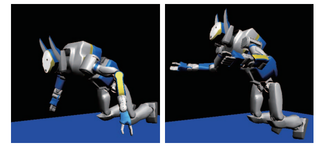

Fig. 10. The averages of falling states that could transfer to state #6(left) and could not because of overload(right) respectively

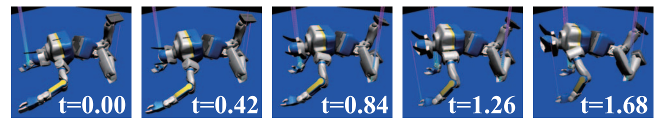

Fig. 11. Example of failed transitions(Unit:[s])

overload(right). The main difference between these states is joint angles of shoulders. When a transition starts from a posture raising arms, the arms collides with the floor, push up its body and suffer from heavy load while transferring to state #6. Actually, 80[%] of the overload was detected at arm joints. Figure 11 shows an example of failed transitions to state #6. In this case, an overload of a wrist joint of the right arm was detected.

Although the collision-free motion generation method is used, there are 12 cases where *Collision* occurs in 36,000 simulations. They are categorized into two cases. One is cases where there is no collision-free path between a lying state and the states in the graph. The other is cases where $\delta\boldsymbol{q}$ calculated by Fig.6 is canceled by an original transition motion.

Next, the target selections and transition trials are executed using 1,000 lying states used in Section II-D. Table V shows results which correspond to Table II. A selection ratio to state #11 increased since falling states which can transfer to state #11 increased. As *Collision* almost never happens, *Success* ratios are improved.

The total success ratio became 62.5[%] which is about 10[%] higher than the result in Section II-D. And the ratio of falling states which can transit to any of six known states became 72.5[%] which is about 15[%] higher than the result in Section II-D by changing a transition motion generation method and a judgment method.

Table VI shows results of the transitions from lying states which can transfer to one of six states as Table III. In this case, the total success ratio is 86.2[%] and it is almost same in the case of Section II-D.

Figure 12 shows examples of the transitions. A lying state in the transition to state #12 looks similar to state #10

TABLE IV
RESULTS OF TRANSITION TRIALS(UNIT:[%])

State No.	success	failure	collision	overload
5	3.2	55.6	0.0	41.1
6	7.9	47.1	0.0	41.1
7	9.8	52.1	0.0	38.1
10	5.1	49.7	0.1	45.2
11	46.6	13.1	0.0	40.3
12	52.0	9.9	0.0	38.1

TABLE V
RESULTS OF SELECTION AND TRANSITION TRIALS(UNIT:[%], ALL FALLING STATES)

No.	selected	success	failure	collision	overload
5	8.8	21.6	3.4	0.0	75.0
6	17.0	20.6	1.2	0.0	78.2
7	13.9	29.5	0.7	0.0	69.8
10	2.4	37.5	37.5	0.0	25.0
11	23.3	91.8	0.4	0.0	7.7
12	34.6	88.7	0.6	0.0	10.7

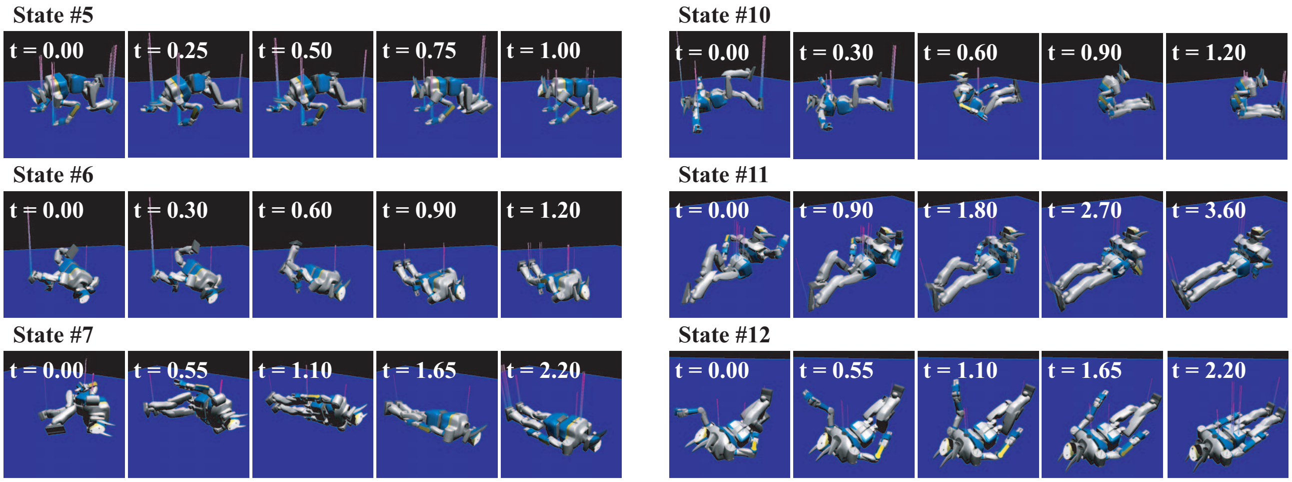

Fig. 12. Examples of transitions(Unit:[s])

TABLE VI

RESULTS OF SELECTION AND TRANSITION TRIALS(UNIT:[%], ONLY FOR FALLING STATES WHICH CAN TRANSIT TO KNOWN STATES)

No.	selected	success	failure	collision	overload
5	3.6	73.1	0.0	0.0	26.9
6	9.4	51.5	1.5	0.0	47.1
7	7.7	73.2	1.8	0.0	25.0
10	3.2	39.1	39.1	0.0	21.7
11	31.6	93.4	0.4	0.0	6.1
12	44.6	95.2	0.3	0.0	4.6

because it is moving up its upper body on the first glance. And a falling state in a transition to state #10 looks similar to state #12 because it is lying on the back. It is interesting that different states are selected and those transitions succeed actually. The transitions to state #7 and #10 may produce large impact forces when legs or the body contact the ground. To plan feasible motions on a real robot, several evaluation criteria must be added to the acquisition process of $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$.

IV. SUMMARY AND CONCLUSION

This paper aims to realize the getting up ability which can start from an arbitrary lying state on the flat floor and the proposed method (1)determines the degree of similarity between the current falling state and predefined falling states using Mahalanobis distance, (2)generates a collision-free motion to the most similar state, and (3)plans a sequence of motions using a state transition graph.

The simulation results show that a correct target state can be selected with 85[%], provided a lying state can transfer to any of the state in the graph. Actually, about 30[%] of lying states did not meet this condition and the total success ratio was about 60[%]. In order to improve the ratio, the lying states which can not to transfer to any state must be decreased. To this end, a new transition motion generation method is required which remove heavy load on joints caused by contacts with the ground. Future works include the development of this method and a strategy which accepts environments which have slopes and terrains.

REFERENCES

[1] M. Hirose, Y. Haikawa, T. Takenaka, and K. Hirai. Development of Humanoid Robot ASIMO. In *Int. Conference on Itelligent Robots and Systems, Workshop2*, 2001.

[2] Y. Kuroki, M. Fujita, T. Ishida, K. Nagasaka, and J. Y amaguchi. A Small Biped Entertainment Robot Exploring Attractive Applications. In *Proc. of the 2003 IEEE International Conference on Robotics & Automation*, pp. 471–476, 2003.

[3] Kenji KANEKO, Fumio KANEHIRO, Shuuji KAJITA, Masaru HIRATA, Kazuhiko AKACHI, and Takakatsu ISOZUMI. Humanoid Robot HRP-2. In *Proc. of the 2004 IEEE International Conference on Robotics & Automation*, pp. 1083–1090, 2004.

[4] J.Morimoto and K.Doya. Acquisition of standup-up behavior by a real robot using hiera rchical reinforcement learning. *Robotics and Autonomous Systems*, Vol. 36, No. 1, pp. 37–51, 2001.

[5] M.Inaba, F.Kanehiro, S.Kagami, and H.Inoue. Two-Armed Bipedal Robot that can Walk, Roll-over and Stand up. *Proc. of Int. Conf. on Intelligent Robots and Systems*, pp. 297–302, 1995.

[6] Takayuki Furuta, Yu Okumura, Tetsuo Tawara, and Hiroaki Kitano. 'morph': A Small-size Humanoid Platform for Behavior Coordination Research. In *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, pp. 165–171, 2001.

[7] Fujitsu Automation Limited. `http://www.automation.fujitsu.com/group/fja/services/hoap/`.

[8] Y. Kuroki, T. Ishida, J. Yamaguchi, M. Fujita, and T. T. Doi. A Small Biped Entertainment Robot. In *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, pp. 181–186, 2001.

[9] Y. Kuniyoshi, G. Cheng, and A. Nagakubo. Etl-humanoid: A research vehicle for open-ended action imitati on. In *Int. Symp. on Robotics Research*, Vol. 1, pp. 42–49, 2001.

[10] Y.Kuniyoshi, Y.Ohmura, K.Terada, and A.Nagakubo. Dynamic Roll-and-Rise Motion by an Adult-Size Humanoid Robot. *Journal of the Robotics Society of Japan*, Vol. 23, No. 6, pp. 706–717, 2005.

[11] F.Kanehiro, K.Kaneko, K.Fujiwara, K.Harada, S.Kajita, K.Yokoi, H.Hirukawa, K.Akachi, and T.Isozumi. The First Humanoid Robot that has the Same Size as a Human and that can Lie down and Get up. In *Proc. of the 2003 IEEE International Conference on Robotics & Automation*, pp. 1633–1639, 2003.

[12] L.E.Kavraki, P.Svestka, 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.

[13] F.Kanehiro, M.Inaba, H.Inoue, and S.Hirai. Developmental Realization of Whole-Body Humanoid Behaviors Based on StateNet Architecture Containing Error Recovery Functions. In *Proc. of the First IEEE-RAS International Conference on Humanoid Robots*, 2000.

[14] Fumio Kanehiro, Hirohisa Hirukawa, and Shuuji Kajita. OpenHRP: Open Architecture Humanoid Robotics Platform. *The International Journal of Robotics Research*, Vol. 23, No. 2, pp. 155–165, 2004.

[15] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-Tree:A Hierarchical Structure for Rapid Interference Detection. In *Proc. of ACM Siggraph '96*, 1996.