Robotica (2009) volume 27, pp. 249–257. © 2008 Cambridge University Press
doi:10.1017/S0263574708004645 Printed in the United Kingdom

Fall detection in walking robots by multi-way principal component analysis

J. G. Daniël Karssen* and Martijn Wisse

Department of Mechanical Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands.

(Received in Final Form: April 3, 2008. First published online: May 8, 2008)

SUMMARY

Large disturbances can cause a biped to fall. If an upcoming fall can be detected, damage can be minimized or the fall can be prevented. We introduce the multi-way principal component analysis (MPCA) method for the detection of upcoming falls. We study the detection capability of the MPCA method in a simulation study with the simplest walking model. The results of this study show that the MPCA method is able to predict a fall up to four steps in advance in the case of single disturbances. In the case of random disturbances the MPCA method has a successful detection probability of up to 90%.

KEYWORDS: Bipeds; Legged robots; Humanoid robots; Robot dynamics; Pose estimation and registration.

1. Introduction

All two-legged (bipedal) walking systems, humans and robots alike, will occasionally encounter disturbances that are so large that a fall is inevitable. To minimize damage, the upcoming fall must be detected and a proper "bracing" action must be taken. Or, in a milder form, perhaps the fall detection can activate a (costly) emergency response which can still prevent the fall, see Fig. 1. Irrespective of the type of reaction, this paper focuses on the detection part only.

This paper presents a new method for fall detection in bipedal robots. It seems that no such method exists yet, and that it is unknown how the human brain performs this type of pattern recognition. In the development of the new method, we assume that the robot has full state information (all relevant positions, angles, and velocities).

Our interest in fall detection arises from our research into limit cycle walking.[1] With this concept, base on passive dynamic walking,[2] we build bipedal robots with unprecedented low energetic cost of transport.[3] The early limit cycle walking prototypes were optimized for low energy usage and consequently they had little resistance against disturbances. We are now building bipeds with more actuation capabilities and better robustness.[4–7] Motions that would have resulted in a fall can now be counteracted at the cost of some extra energy usage. That is why a fall detection algorithm has our interest. Nevertheless, we are convinced that such an algorithm will be useful for all biped walking systems using any type of stability control, such as the well known "Zero Moment Point" based control.[8,9] At the very least, the algorithm can sound an alarm and alert the robot researcher to catch the robot.

All bipedal robots are complex dynamic systems and this gives a couple of challenges for the monitoring of a walking robot. The main challenges are:

High dimensionality. Modern walking robots have up to 32 degrees-of-freedom (DoF)[10] and each DoF represents two states (position and velocity). The high dimensionality can result in large computational times or memory requirements due to the "curse of dimensionality". For example, a lot of tests can be required to test for every possible state if the robot is going to fall or not. For a system with M state variables and N grid points at each dimension are N^M tests required.

Not steady state. Walking is a cyclic motion, so the system goes through a sequence of states each cycle. It is therefore not possible to monitor the system as a steady-state system. Thus, one cannot use fixed limits on the state variables for all detections.

Different units. The state variables have different units, for example meters for position and radians per second for angular velocity. This difference in units causes problems when calculating Euclidean distances in state space. A problem can be that one state variable has more effect than another. To remove this effect the state variables can be scaled.

No fixed cycle duration. The motion of a walking robot is quasi-periodic, but it can have variation on the period. Due to this variation it is not possible to compare cycles with a fixed time scale.

Discontinuous state transitions. In walking there is at least one impact every step, when the swing leg hits the ground. During an impact, the velocities change rapidly and this causes a jump in state space.

Nonlinear behavior. The dynamics of a walking robot are highly nonlinear. This means that linear analyzing methods can not be used before the nonlinearity in the data is removed.

In this paper we present in this paper the multi-way principal component analysis (MPCA) detection method for fall detection in bipedal robots. Nomikos and MacGregor[11] introduced the MPCA method for the monitoring of batch processes. The monitoring of batch processes has similar problems as the monitoring of walking robots. A batch

* Corresponding author. E-mail: j.g.d.karssen@tudelft.nl

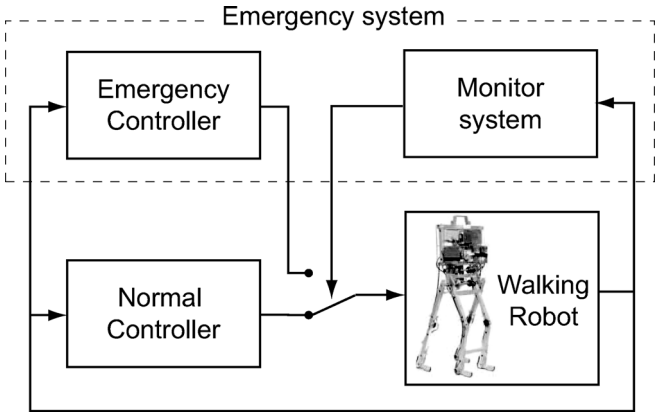

Fig. 1. Normal controller with added emergency system to prevent falling when encountering large disturbances.

process is, just like a walking robot, a nonlinear cyclic process and has a high number of state variables with different units. Since its introduction, the MPCA method is successfully implemented in industrial applications[12] and a quite a few variations on the MPCA method have been introduced. The large variety of successful implementations shows the success of the MPCA method. This success and the similarities between batch processes and walking robots show the potential of the MPCA method for walking robots.

The goal of this paper is to study the performance of MPCA detection method for fall detection in walking robots. We study the performance with a simulation. The model used in this simulation is described in Section 3. But first we discuss, in Section 2, how the MPCA monitoring method works and how it can be implemented for walking robots. The results of the simulation study are shown in Section 4. These results are discussed in Section 5 and conclusions are drawn in Section 6.

2. MPCA Monitoring

The main assumption behind MPCA monitoring is that the probability for failure can be estimated by the error of the current trajectory with respect to the nominal trajectory scaled by the nominal variation around the trajectory. The scaling with the nominal variation is done so that a larger error is allowed in parts of the cycle that have large variation during normal operation. For this scaling only the relative shape of the variation is important and not the absolute value. Before the monitoring can take place, the nominal trajectory and the nominal variation around this trajectory have to be found. This is done with a set of "good" cycles. These cycles are generated by running the system for a number of cycles with small disturbances. The small disturbances are large enough to create variation between the cycles but are small enough to not create a failure.

In the remainder of this section we will look further into the MPCA monitoring. In Section 2.1 the origin of the MPCA monitoring method and the different modeling methods are discussed. In the next section we show how one of these modeling methods can create a model out of the data from the "good" cycles. This model is used in Section 2.3 to construct a measure for monitoring. And in the final section of this paper we will discuss a couple practical issues.

2.1. Origin of MPCA monitoring

The MPCA method for monitoring presented in this paper is based on a monitoring method by Nomikos and MacGregor.[11] They introduced MPCA monitoring for the monitoring of batch processes in the production of chemicals. In a batch process, a reactor is loaded with raw materials and a reaction takes place. After the reaction is finished, the product is collected and the reactor can be loaded for the next batch. This cyclical process has to be monitored to ensure safe operation and to get consistent high-quality products. The monitoring method of Nomikos and MacGregor predicts if the quality of a batch is within a normal quality range.

For the monitoring, a model of the normal behavior of the system is used. This model is obtained from a set of "good" cycles. There are a couple of ways to construct this model. Nomikos and MacGregor used the global modeling method. This means that they used one model to describe the whole trajectory. Opposite to the global modeling method is the local modeling method.[13] The local method uses a model for each point in the trajectory. The main difference between monitoring with the global model and the local model is in the measurements that are used. With the local model only the current measurements are used to predict failure. With the global model all the measurements since the start of the run are used. Besides local and global modeling there are other methods suggested in literature. These are methods that are between the local and global modeling method like adaptive modeling,[14] time evolving modeling[13] and variable-wise unfolding.[15]

The choice for a modeling method depends on the amount of state information that can be measured. With our walking robots the complete state can be measured. With the complete state available at every moment there is no need for using other measurements than the current measurements and so the local modeling method is best suited.

2.2. Modeling

With the local modeling method, a model can be constructed that consists of the nominal trajectory and the nominal variation around the nominal trajectory. This model is constructed with a set of "good" cycles. This set consists of I trajectories in J-dimensional space.

2.2.1. Slicing the trajectories. We like to analyze the trajectories with the principal component analysis (PCA) method. PCA is a data analyzing method that removes the linear correlation between the variables by rotating the coordinate system. It is impossible to analyze the trajectories directly with the linear PCA method because the trajectories can be very nonlinear (Fig. 2A). To remove the nonlinearities we look only at the position in state space with respect to the nominal trajectory. This is done by slicing the trajectories perpendicular to the nominal trajectory with K planes (Fig. 2B). These planes are $(J-1)$-dimensional, one dimension lower than the space of the trajectories. The planes can be seen as cross sections of the trajectories. Each of these

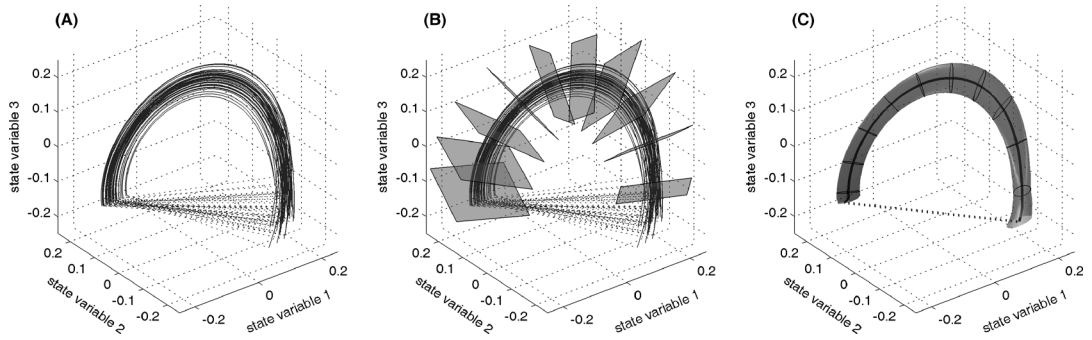

Fig. 2. (A) A set of 50 nonlinear trajectories in a three-dimensional state space. The dotted lines indicate that the trajectories make a jump in state space. (B) Planes slice the trajectories perpendicular. The points in these planes can be analyzed without the nonlinearities of the trajectories. (C) With the points in the planes the average trajectory and the boundary is estimated. The boundary in a plane is an ellipse. The ellipses of all the planes combined results in a tube. Note that the shown tube is created with more planes than are shown.

cross sections separately can be analyzed with PCA because the nonlinearity of the trajectory is removed.

A problem with this slicing perpendicular to the nominal trajectory is that the nominal trajectory is not known (yet). The nominal trajectory can be estimated with the average of the "good" trajectories in time. The time of each of the trajectories is normalized with its cycle duration to prevent problems with differences in cycle duration.

The slicing planes are equally spaced over the nominal trajectory. The measurement samples of the trajectories will never lie exactly on these planes. We use a linear interpolation between the two closest samples to find the intersection of the trajectory with the slicing plane.

2.2.2. Analyzing the slices. Each of the K slices consist of I points on a $(J-1)$-dimensional plane in a J-dimensional space. For each of these slices we want to find two things: the average point and the boundary of the cloud of points. The average points of all the K slices will be used as an estimation of the nominal trajectory and the boundary as an estimation the variation with respect to the nominal trajectory. For the normal trajectory we could also use the fixed point trajectory instead of the average trajectory. However, we will only use the average trajectory, because in a physical system the fixed point trajectory cannot be determined due to the inevitable presence of noise.

The average point, for each slice, is found by taking the average over all the points for each of the J dimensions separately,

$$\bar{\mathbf{X}}_j^k = \frac{1}{I}\sum_{i=1}^{I}\mathbf{X}_{ij}^k \qquad \text{for } j = 1,\ldots,(J-1), \tag{1}$$

in which $\mathbf{X}_{ij}^k$ is the jth coordinate of point i and $\bar{\mathbf{X}}_j^k$ is the jth coordinate of the average point. The index k refers to the slice k.

The boundary can also be estimated for each dimension j separately, but this will result in large errors if the dimensions are correlated (Fig. 3). Therefore we will first apply a PCA to remove all the linear correlation. The PCA rotates the coordinate axes in such a way that the correlation between variables is placed on a single rotated axis. Before the rotation of the axes, the data is mean-centered so that the average point

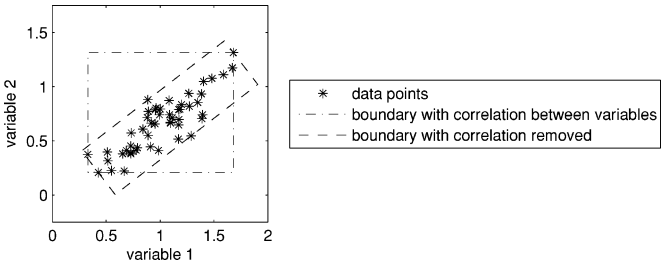

Fig. 3. A random set of points with linear correlation between the two variables. The Min–Max boundary determined for the two variables separately results in a large error due to the correlation. With the correlation removed the error is much smaller.

does not move when rotating the axes. In matrix notation the rotation of the coordinate system can be written as

$$\mathbf{T}^k = \mathbf{P}^k(\mathbf{X}^k - \bar{\mathbf{X}}^k) \tag{2}$$

in which $\mathbf{T}^k$ is the data in the new coordinate system and $\mathbf{P}^k$ the transformation matrix. For a detailed description of the PCA see ref. [16].

We assume that the points have a nominal distribution after the correlation is removed. This assumption is based on the fact that a linear combination of random variables tends toward a normal distribution.[17] A normal distribution does not have a boundary, but a 95% interval can be found by taking two times the standard deviation in both directions. The standard deviation σ for each of the uncorrelated dimensions is

$$\sigma_j^k = \frac{1}{I-1}\sum_{i=1}^{I}\mathbf{T}_{ij}^k \qquad \text{for } j = 1,\ldots,(J-1). \tag{3}$$

The normal distribution for each of the dimensions results in a multivariate normal distribution. For this multivariate normal distribution the 95% interval is an ellipsoid. This $(J-1)$-dimensional ellipsoid has its axes on the rotated axes after PCA and can be rotated back into the original coordinate frame (Fig. 4).

The ellipsoids found for the K slices can be combined in a tube around the nominal trajectory. Figure 2C shows an example of such a tube for a three-dimensional system. The

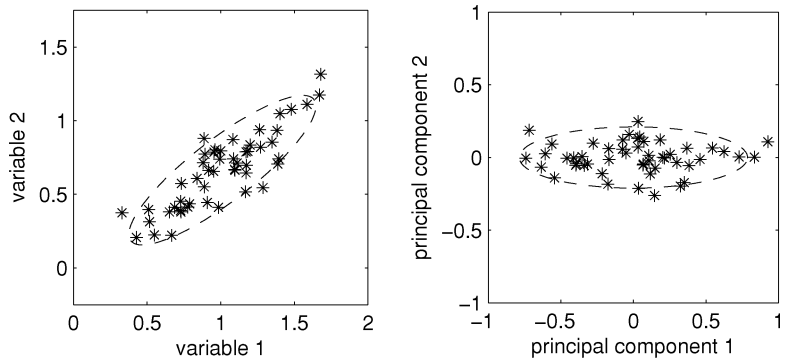

Fig. 4. A random set of points (the same as in Fig. 3). With PCA the coordinate system is rotated so that linear correlation between the variables is removed. With the correlation removed the boundary of the multivariate normal distribution is estimated with an ellipse.

tube is the boundary of the "good" cycles and shows the nominal variation around the nominal trajectory. In the next section we will discuss how this tube can be used for the monitoring of the system.

2.3. Monitoring

In the last section, we estimated the nominal trajectory and the nominal variation around this trajectory with the trajectories of a set of "good" cycles. In this section, we will use the nominal trajectory and the nominal variation to monitor the system. For the monitoring we look at the error between the current state of the system and the nominal trajectory and scale this error by the nominal variation. The scaling is done so that on parts of the trajectory where the "good" cycles had a lot of variation the error is allowed to be bigger than on parts with small variations. If the scaled error becomes above a set limit, the monitoring system thinks that the failure change is too high and an emergency action is applied.

2.3.1. Slice selection. The first step in determining the error is to determine to which of the modeling slices the current state belongs. This is determined with two measures, the distance between the current state and the slice and the distance between the current state and the intersection between the slice and the nominal trajectory. The first measure indicates how close the current state is to the plane. The second measure is added to prevent that a slice is selected that intersected the nominal trajectory far from the current state (Fig. 5).

2.3.2. D-statistic. With the slice selected, the next step is to determine the error between the current state and the nominal trajectory. This is done as follows:

$$e_j = x_j - \bar{\mathbf{X}}_j^k \qquad \text{for } j = 1, \ldots, (J-1), \tag{4}$$

in which x_j is the jth coordinate of the current state, $\bar{X}_j^k$ is the jth coordinate of the intersection between the selected slice k and the nominal trajectory and e_j is the error in dimension j. The error vector e has to be scaled. The scaling is done in the rotated coordinate system of the slice. To get the error in the rotated coordinate system, the error vector e is multiplied with the rotation matrix $\mathbf{P}$ of the selected slice k,

$$\tilde{e} = \mathbf{P}^k e. \tag{5}$$

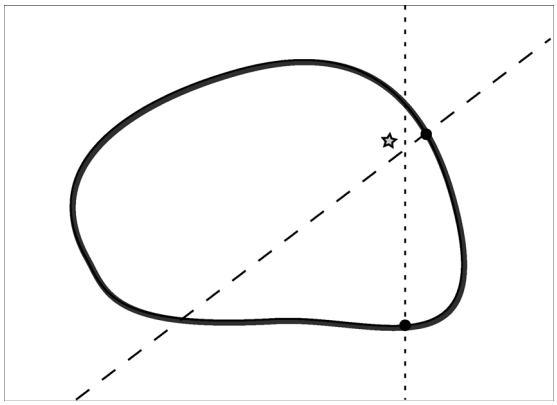

Fig. 5. A trajectory (solid line) in a two-dimensional state space with two slices (dashed and dotted lines). The current state (star) has equal distance to both slices. The dashed slice is selected, because it has the shortest distance between the current state and the intersection slice-trajectory (dot).

The rotation matrix $\mathbf{P}$ is the same as the rotation matrix used in equation (2) to remove the correlation between the good trajectories. The error $\tilde{e}_j$ for each rotated dimension is scaled by dividing it with the standard deviation σ^k of the "good" set

$$\check{e}_j = \frac{\tilde{e}_j}{\sigma_j^k} \qquad \text{for } j = 1, \ldots, (J-1). \tag{6}$$

By taking the mean of squared error vector $\check{e}$ we get the D-statistic,

$$D = \frac{1}{J-1} \sum_{j=1}^{J-1} \check{e}_j{}^2. \tag{7}$$

This D-statistic gives how far the current state is off the nominal trajectory with respect to the variation in the set of "good" cycles. So it is a measure for how well the current state fits in the normal behavior of the system.

2.3.3. Limit. The next thing we want to know is how far the system can deviate from its nominal trajectory before it is going to fail. In other words, above which value for the D-statistic is an emergency action required. The setting of maximal value of the D-statistic $D_{\max}$ depends on the size and frequency of the expected disturbances, because the

probability depends, besides on the current state, also on the future disturbances.

There is also a trade-off in the maximal value of the D-statistic between an early detection and as few as possible unnecessary actions. Early detection of a system failure is necessary to have enough time to take an emergency action. Early detection can be achieved by a low $D_{\max}$. A disadvantage of a low $D_{\max}$ is that it can cause unnecessary emergency actions. A solution for this trade-off is to set the $D_{\max}$ as high as possible while ensuring that there is enough time for the emergency action.

These two aspects show that the choice of the $D_{\max}$ depends on the system properties, such as emergency action time. Because of the dependency on the system properties it is not possible to give a general setting of the maximal value of the D-statistic. In Section 4 we will discuss how to set the maximal value for a walking robot.

2.4. Practical issues

2.4.1. Scaling. The state variables can be measured using a variety of units. A difference in used units for the state variables can cause a difference in the D-statistic. This difference is caused by the fact that a slicing plane that is perpendicular to a trajectory for one set of units does not have to be perpendicular for another set of units. To eliminate this effect, the state variables are scaled to get dimensionless units. The most common way of scaling is to scale mass with the total mass of the system, length with the leg length, and time with the nominal step time.

2.4.2. Discontinuous trajectories. The trajectory of a system can have discontinuities, for example an instantaneous change in the velocity caused by an impact. These discontinuities cause problems for the slicing of the set of "good" trajectories. A discontinuity in a trajectory is a jump in state space. This jump can cause that not all of the trajectories are sliced by a slicing plane, because a trajectory can pass the plane by the jump in state space. If not all the trajectories are sliced there will be less data points in the plane. The remaining data points will not have a normal distribution, because the trajectories that are not sliced are not a random selection out of all the trajectories. This is not random, because the place of the discontinuity is a function of the state. Without a normal distribution, the boundary of the points can not be estimated with the standard deviation. If in this case, the boundary of the points is estimated with the standard deviation, the D-statistic will be incorrect. An incorrect D-statistic due to a discontinuity can trigger an unnecessary emergency action. To prevent this, the D-statistic will be ignored when the selected plane is passed by one of the "good" trajectories.

2.4.3. Number of "good" trajectories and number of slices. When applying the MPCA monitoring method there are two choices that have to be made: the number of "good" trajectories I used for the modeling and the number of slices K used for the slicing of the trajectories. For both of these choices, it applies that the higher the number, the more accurate the method becomes. In practice, there are time and memory restrictions. There are two time restrictions, one on the time used for the generation of the model and one on the time used during monitoring. The time for the model generation is only limited by the time the researcher is willing to spend on it. The major part of this time is used to generate the "good" trajectories. The time used during monitoring is restricted by the available computing time and the monitoring frequency. To give an indication of the processing time: the calculation of the D-statistic in a system with 4 state variables and 1000 slices takes about 0.3 ms in a MATLAB environment on a 1.66 GHz Intel Core Duo processor. The number of slices is mainly limited by a memory constraint. The memory required is linear related with the number of slices. The data of each slice consists of $(J + J^2 + (J - 1))$ numbers, to represent respectively the center point, the orientation of the coordinate system, and the standard deviations, with J the number of state variables. In case of a system with 8 state variables, 1000 slices, and single precision numbers, the total memory required is 0.63 MB.

3. Test Method

The ability of the MPCA monitoring method to predict the fall of a walking robot is tested on a model of a walking robot. On this model a set of disturbances are applied and the D-statistic is compared with the behavior of the model. The next sections describe the model and the set of disturbances.

3.1. Model

The model used for this study is the simplest walking model by Garcia *et al.*[18] This two-dimensional model consists of two rigid links, which are connected at the hip. There are three point masses in the model, one point mass at the hip and two infinitesimally small masses at the feet. The lengths and masses are scaled with the leg length and hip mass to get dimensionless units. The feet are point feet and the impacts of the feet with the ground are modeled as fully plastic collisions. Foot scuffing at mid stance is ignored so that the swing leg can swing from the rear to the front. The model has no control or actuation and it gains energy by walking down a slope of 0.004 rad.

The model has two DoF, the stance leg can rotate with respect to the floor around its point foot and the swing leg can rotate around the hip. The two DoF of this second-order system results in a four-dimensional state space. We define the four state dimensions as the angle and angle rate for both the stance leg and the swing leg, $[\phi_{st}, \dot{\phi}_{st}, \phi_{sw}, \dot{\phi}_{sw}]$ (Fig. 6).

It has been shown that the simplest walking model can walk in a stable cycle on a flat surface and can overcome small disturbances.[5] The largest single step-down it can overcome has a height difference of 0.13% of its leg length. A higher step-down will result in a state at which the deviation of the nominal trajectory becomes larger every step and finally results in a fall.

3.2. Disturbances

In this study we use floor irregularities as disturbance source. Floor irregularities are the most common disturbances for walking robots. The irregularities are implemented as a height difference in the floor. This height difference can be positive or negative resulting in a step-down or a step-up. Two types of disturbance patterns are used: a single

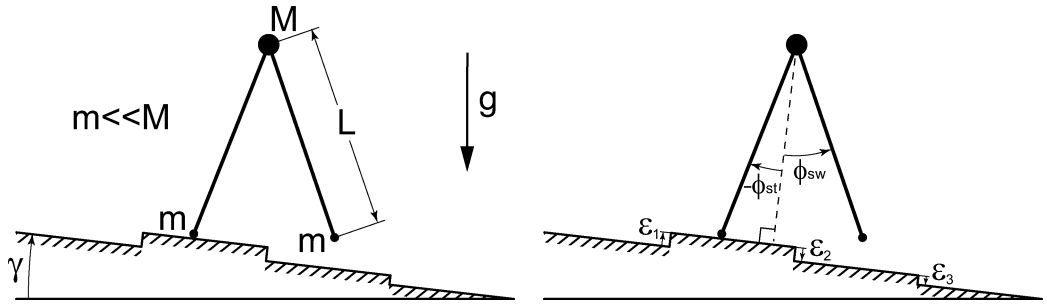

Fig. 6. The simplest walking model walking down a slope γ of 0.004 rad. The model is disturbed by random floor irregularities.

impulsive disturbance and a Gaussian disturbance pattern. The impulsive disturbance is a single step-down followed by disturbance-free floor. The Gaussian disturbance pattern is a randomly generated floor with step-ups and step-downs having a Gaussian distribution with a zero average and a standard deviation of σ.

For the estimation of the nominal trajectory and the variation, a set of "good" cycles is required. This set is generated by a single test run of 50 steps with the Gaussian disturbance pattern with a standard deviation σ of 0.02%. At this disturbance level the model walks on average 61 steps without failures, so the generation of the 50 "good" cycles is not a problem. For the slicing of trajectories, 1000 planes are used. This amount of slices gives an accurate model with a memory use of about 0.3 MB.

4. Results

Figure 7 shows the D-statistic of two test runs during a couple of steps. At the beginning of the third step ($t = 7.6$ s) a step-down disturbance is applied. The height difference of step-downs are 0.13% and 0.14% of the leg length for respectively the first and second test run. The simplest walking model is able to keep on walking after the disturbance of 0.13%, but falls down after the disturbance of 0.14%. For both runs the D-statistic shows an impulsive increase at the moment the disturbance was applied. After the impulsive increase, the D-statistic for the first run decreases and after a couple of steps the D-statistic is back at the original level. The D-statistic for the second run shows a different behavior, it keeps increasing after the first impulsive increase. Three steps after the disturbance ($t = 15.6$ s) the model starts falling and the D-statistic rapidly increases. At $t = 18.7$ s the model lays on the floor with a D-statistic of about 10^6.

Figure 8 shows the D-statistic of 20 test runs on three irregular floors. The floor irregularities have a Gaussian distribution with a zero average and a standard deviation of 0.020%, 0.025%, and 0.030% of the leg length for respectively the first, second, and third test set. In each of the test runs the disturbances cause the model to fall. In the figure, the time axes are shifted to synchronize the falls. The D-statistic shows similar behavior for the three floors. Until approximately 20 s before the fall, the D-statistic is more or less constant with an average of around 2. In the last

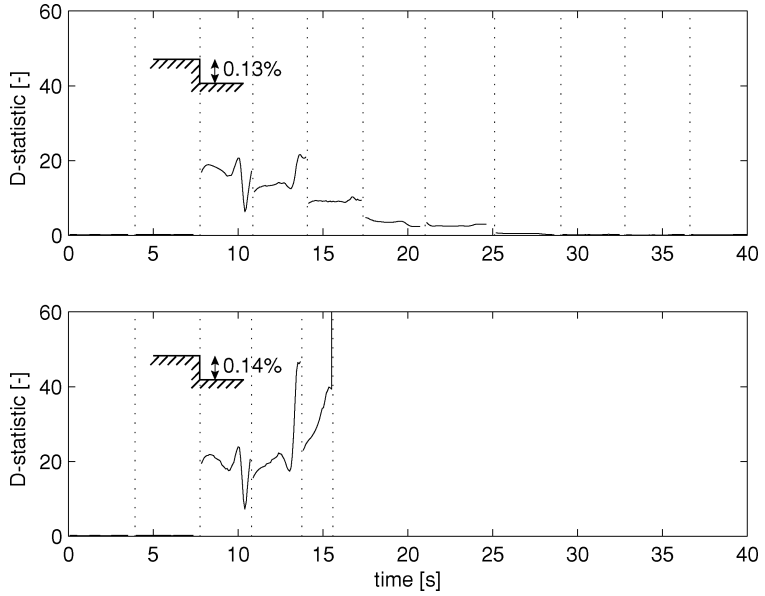

Fig. 7. The D-statistic in time for two test runs with a disturbance at $t = 7.6$ s. In the first run (top) the disturbance is so small that the model can recover. In the second run (bottom) the disturbance is slightly larger and causes the model to fall. The dashed vertical lines indicate stance leg transition. Note the gaps in the D-statistic graph. At these gaps, the D-statistic cannot be used and is therefore ignored, as explained in Section 2.4.2.

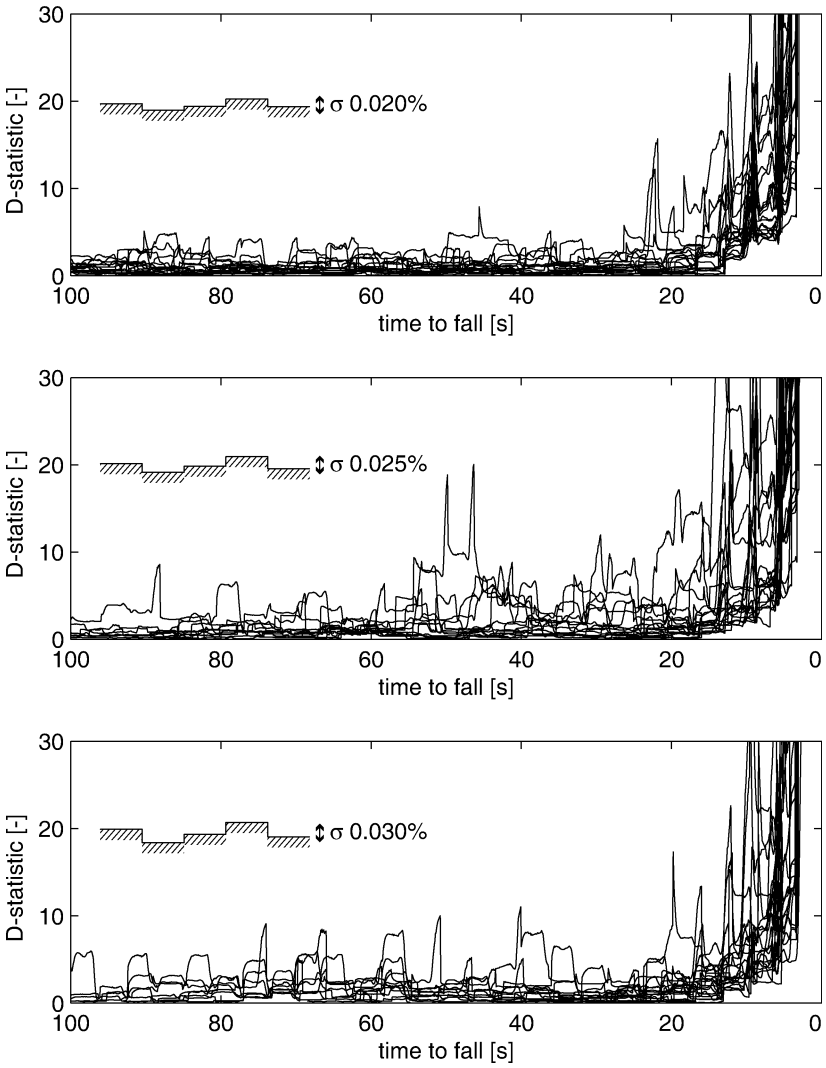

Fig. 8. The D-statistic of 20 test runs on three irregular floors. The floor irregularities have a Gaussian distribution with a zero average and a standard deviation of 0.020%, 0.025%, and 0.030% respectively for the first, second, and third test set. The runs are shifted in time so that they all end at $t = 0$ in a fall of the model.

four steps before the fall, approximately between $t = 20$ and $t = 0$, the D-statistic of all the runs start increasing and reach values up to 10^6 at $t = 0$. There is a difference between the D-statistic of the three floors in the amount and the height of the spikes in the part before the last 20 s. The D-statistic of the floor with a standard deviation of 0.020% has less high spikes then the two other test sets. Another difference is the minimal D-statistic during the last 10 s. This minimal D-statistic increases the fastest for the 0.020% test run and the slowest for the 0.030% test run.

5. Discussion

The MPCA monitoring method was introduced to predict the fall of a walking robot, so that an emergency action can be taken to prevent the fall. To predict a fall the monitoring system should be able to distinguish between states that lead to a fall and states that lead to a walking cycle. The area of all the states that lead to a walking cycle is called the basin of attraction (BoA). The MPCA monitoring method distinguishes between states with a D-statistic below a maximum $D_{\max}$ and states with a D-statistic above $D_{\max}$.

The relation between the D-statistic and states inside and outside the BoA can be seen in the results of the single step-down tests (Fig. 7). In the second test (step-down of 0.14%) the state of the system outside the BoA after the step-down disturbance, because without anymore disturbances the system fails. This system has a higher D-statistic than the system in the first test run, which state is inside the BoA for the complete run. Ideally, the D-statistic of any state outside the BoA should be higher than the D-statistic of any state inside the BoA. This is not the case in these tests, because the minimum of the part of the second run that is outside the BoA is 7.2, which is lower than the maximum of 21.6 of the first run. But it is still possible to predict the fall in the second run without causing a false alarm in the first run. With a $D_{\max}$ of 22 the fall in the second run will be predicted at $t = 10$ s and this gives more than two steps to prevent the fall.

For walking on an irregular floor the current state information is not enough for predicting falls, because the upcoming irregularities in the floor also affect whether the robot is going to fall or not. For example, if the robot is outside the BoA it can go back into the BoA by a floor disturbance. The floor irregularities are assumed to be unknown, so we

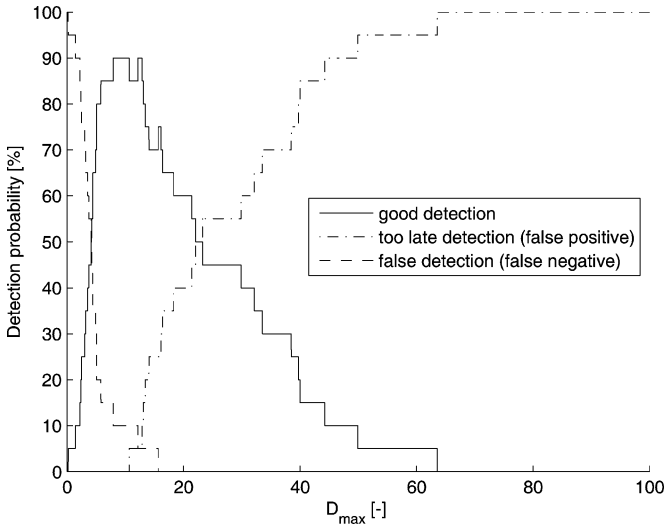

Fig. 9. The limit for the D-statistic $D_{\max}$ vs. the percentage of good, too late, and false detections. The detection probabilities are estimated with a set of 20 test runs (the same as for Fig. 8 (top)). Good detections predict the fall at least one step and maximal four steps before the fall.

can make only a prediction on the probability of a fall. Due to this uncertainty the setting of the $D_{\max}$ will be a trade-off between the amount of false positives and false negatives. A false positive is when a fall is predicted too late and a false negative is when a fall is predicted but does not occur. Figure 9 shows this trade-off based on the data from the 20 test runs with an irregular floor. For this example false positives are counted if the fall was not detected at least one step before the fall. The false negatives are harder to be determined because all the test runs ended with a fall. False negatives are counted if there were more than four steps between the detection and the fall. The four steps limit is taken, because the single step-down test shows that it takes less than four steps to go from a state just outside the BoA to a fall.

The trade-off shown in Fig. 9 has no optimum. It is not always the best to select the $D_{\max}$ value with the highest probability on good detection, because the setting of the $D_{\max}$ depends also on if the false positives or the false negatives should be avoided. For example if a fall has to be avoided at all times, because it causes a lot of damage, no false positives should be allowed. To avoid false positives the $D_{\max}$ should be low. In the example of Fig. 9 the $D_{\max}$ should be maximal 11. The $D_{\max}$ should be at least 18 if false negatives have to be avoided.

The ability of the MPCA method to predict a fall depends on the kind of disturbances that are encountered. Figure 10 shows the detection probability versus $D_{\max}$ for three levels of disturbance. For the lowest disturbance level (standard deviation of 0.020%) the accurate detection probability is up to 90%. So for this disturbance level the MPCA method is good in predicting falls on time. With increasing disturbance levels the maximal good detection probability decreases. This is an effect that is expected, because with higher disturbance levels the system can go faster from a state inside the BoA to a fall. This can also be seen in Fig. 8, where the time from a low D-statistic to a fall is shorter for higher disturbance levels than for lower disturbance levels.

6. Conclusion

In this paper, we introduced the MPCA method for the monitoring of the gait of walking robots. The MPCA monitoring method was introduced to estimate the probability of a fall of a walking robot, so that if a fall is

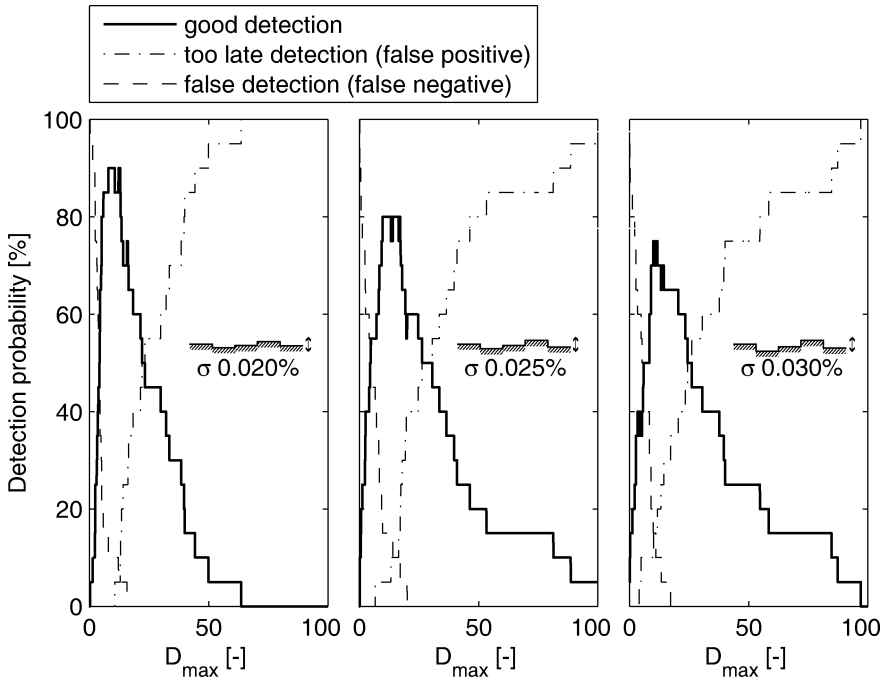

Fig. 10. Same plot as Fig. 9 for three disturbance levels. The lower the disturbance level the better the MPCA method is able to detect a fall on time.

likely to occur an emergency action can be taken. The MPCA method works by comparing the current state of the walking robot to a set of "good" walking cycles. The ability of the MPCA method to predict a fall was studied with the simplest walking model. Out of the results of this study we can draw the following conclusions:

- The MPCA method is able to predict whether the model is going to fall or not, but cannot make an absolute distinction between states inside and outside the BoA.
- In the case of a single disturbance, the MPCA method can predict if the system is going to fall within one step after the disturbance.
- The ability of the MPCA method to predict the fall of the model walking on an irregular floor depends on the size of the irregularities. The smaller the irregularities, the better the predictions.
- The MPCA method can have successful detection probability up to 90% for walking on irregular floor.
- The MPCA method has a low implementation complexity and a low number of test runs is required.

Based on these conclusions, we recommend to use MPCA for fall detection in bipedal robots. We tested the MPCA method on a model of a limit cycle walker, but we expect that the same results holds for other kinds of bipedal robots, such as ZMP walkers. In the near future, we intend to implement this method on a physical prototype. The simulation results presented in this paper suggest that a practical implementation will be successful.

Acknowledgments

The work presented in this paper has been carried out with financial support from the Commission of the European Union, within Framework Programme 6, RTD programme IST, under contract no. FP6-2005-IST-61-045301-STP. The authors would like to thank Frans van der Helm for proofreading.

References

1. D. G. E. Hobbelen and M. Wisse. "Limit Cycle Walking," **In**: *Humanoid Robots, Human-like Machines* (M. Hackel, ed.) Chapter 14. (I-Tech Education and Publishing, Vienna, Austria, 2007).
2. T. McGeer. "Passive dynamic walking," *Int. J. Rob. Res.* **9**(2), 62–82 (1990).
3. S. H. Collins, A. Ruina, R. Tedrake and M. Wisse. "Efficient bipedal robots based on passive-dynamic walkers," *Science* **307**(5712), 1082–1085 (2005).
4. D. G. E. Hobbelen and M. Wisse. "A disturbance rejection measure for limit cycle walkers: The gait sensitivity norm," *IEEE Trans. Rob.* **23**(6), 1213–1224 (2007).
5. D. G. E. Hobbelen and M. Wisse, "Swing leg retraction for limit cycle walkers improves disturbance rejection," *IEEE Trans. Rob.* **24**(2), 377–389 (2008).
6. D. G. E. Hobbelen and M. Wisse, "Ankle actuation for limit cycle walkers," *Int. J. Rob. Res.* (in press).
7. D. G. E. Hobbelen and M. Wisse. "Upper body feedback and feedforward control in limit cycle walkers," *IEEE Trans. Rob.* (in review).
8. M. Vukobratovic, A. Frank and D. Juricic. "On the stability of biped locomotion," *IEEE Trans. Biomed. Eng.* **17**(1), 25–36 (1970).
9. Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki and M. Fujita. "The intelligent Asimo: System Overview and Integration," *Proceedings of International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland (Sep. 30–4 Oct. 2002) pp. 2478–2483.
10. J. Yamaguchi, E. Soga, S. Inoue and A. Takanishi. "Development of a bipedal humanoid robot-control method of wholebody cooperative dynamic biped walking," *Proceedings of 1999 IEEE International Conference on Robotics and Automation*, Detroit, MI, pp. 368–374 (1999).
11. P. Nomikos and J.F. MacGregor. "Monitoring batch processes using multiway principal component analysis," *AIChE J.* **40**(8), 1361–1375 (1994).
12. T. Kourti, J. Lee and J. F. MacGregor, "Experiences with industrial applications of projection methods for multivariate statistical process control," *Comput. Chem. Eng.* **20**, 745–750 (1996).
13. H. J. Ramaker, E. N. M. van Sprang, J. A. Westerhuis and A. K. Smilde, "Fault detection properties of global, local and time evolving models for batch process monitoring," *J. Process Control* **15**(7), 799–805 (2005).
14. J. F. MacGregor, S. Rannar and S. Wold, "Adaptive batch monitoring using hierarchical PCA," *Chemomet. Intell. Lab. Sys.* **41**, 73–81 (1998).
15. P. Nomikos, "Statistical Process Control of Batch Processes", *Chem. Eng. Dept.,* (McMaster University, Hamilton, ON, Canada (1995).
16. P. Geladi and B. R. Kowalski. "Partial least-squares regression: A tutorial," *Anal. Chim. Acta* **185**, 1–17 (1986).
17. G. Hahn and W. Meeker, "Statistical Intervals: A Guide for Practitioners", Wiley, New York (1991).
18. M. S. Garcia, A. Chatterjee, A. Ruina and M. J. Coleman, "The simplest walking model: Stability, complexity, and scaling," *ASME J. Biomech. Eng.* **120**(2), 281–288 (1998).