In Proceedings of the 8th ESA Workshop on Advanced Space Technologies for Robotics and Automation
'ASTRA 2004' ESTEC, Noordwijk, The Netherlands, November 2 - 4, 2004

# Development of a Bioinspired Insect Leg

**Florian Oszwald & Armin Wedler**

Florian.Oszwald@web.de, ArminWedler@gmx.de, Andre.Schiele@esa.int

## ABSTRACT

This document is a short version of the Thesis from Florian Oszwald and Armin Wedler from The University Hanover "Mechantronik Zentrum Hannover". The Thesis have been written under the line of Andre Schiele in the research center of the ESA, the ESTEC, Noordwijk in the Automatisation and Robotic Section in October 2004.

## INTRODUCTION

Robotic explorers play an important role in planetary exploration. Up to now, robots are the only means of gathering detailed in-situ information from the planetary surface of a neighboring planet or the moon. The high risk for men and the high security standards for manned space flight make human exploration difficult.

Many different tasks can be accomplished by extraterrestrial robotic systems. To reach locations of interest apart from the landing site, some kind of locomotion is necessary for these systems. Quite recently two mobile robots have been sent to Mars and perform very successfully. These autonomous rovers are wheeled and they analyze the surface of Mars up to the time of writing. Wheeled rovers are very good in carrying high loads. But when it comes to navigation in rough terrain, they may not perform that well. To prevent the rover from being damaged by collision or from maneuvering it into a deadlock, time consuming computations have to be made.

However, the rover's ability to move in various terrains makes them very useful in many different areas, despite the complex computations. In general, mobile robots exist in varying implementations which not only differ in the way their drive propulsion systems are realized. Some of them have three, four, six or eight wheels; others have chain systems. So there are various possibilities to construct these robots.In nature, many lifeforms are legged. Legs, compared to wheels, are more helpful in unknown rough terrain.

The second aspect is the way robots have been built up till now. Almost every robot has joints with bearings, straight limbs and no natural compliance. In this work, a bioinspired design approach was taken to develop a leg of a hexapod walking robot. Therefore, different insect legs were studied to find appropriate leg dimensions.

A simulator was developed to find opening angles for the joints and to investigate the geometry of the robot. Furthermore, a trajectory planning was accomplished and different gaits were implemented. A complete leg was designed which has adjustable mechanical compliance. Additionally, a new type of joint is proposed here.

## MODELING OF A SIX LEG ROBOT

Since a six-legged robot is a rather complex system, it is convenient to have a simulator. With its help, it is feasible to investigate the robot's walking abilities on different terrains.

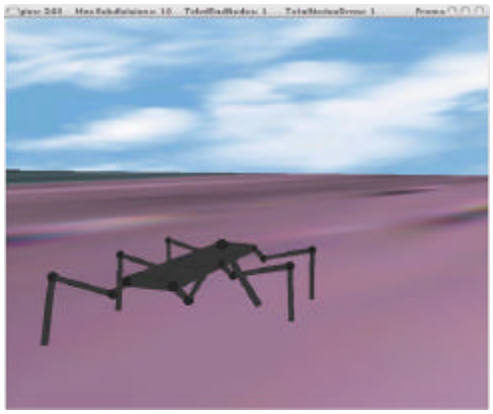

**Figure 1:** Simulation

But before that, other aspects have to be examined. Therefore, dimensions and torques as well as information about joint angles can be obtained in a simulation environment. Furthermore, controllers can be tested in order to work properly and not to destroy the real hardware. In this virtual environment, it is possible to examine different paths and trajectories for the locomotion of the walking robot. Along with this, the correct coordination of the six legs can be explored to achieve different gaits.

With the help of the simulation it is also possible to pay regard to geometrical facets. An important thing to consider is the interference of legs with other parts of the robot throughout the movement.

For the development of the robot leg, different aspects need to be considered. Dimensions of the limbs need to be found. Angles for zero position and angle ranges have to be calculated. Within the simulation, occurring moments and forces can be computed. To obtain these, the kinematic equation has to be found and implemented in the simulation environment. How this is done is described in this chapter.

### Simulation with C

The simulator itself was programmed first in MATLAB and later with C/C++ and OpenGL. MATLAB is briefly introduced, below.

The visualization capabilities of MATLAB with the robotics toolbox proved to be more and more difficult for six legs. Thus the advantage of having some predefined function decreased and so a simulator was programmed using C/C++.

Since some mathematical calculations need to be performed within the simulator, a convenient way for matrix use in the C code had to be found. After some investigation, the matrix template Matrix TCL Lite was chosen [TECH]. With this template class, matrices can be used in C++ just like any other data type. To do so, one has to call a typedef from the C++ code. Templates allow, for example, double type matrix, a float type matrix or an integer type matrix without changing the code. Only three different typedefs need to be done. In this case, only the double type of the matrix is needed. So the call is:

typedef matrix<double> DMatrix;

Where DMatrix is now the new data type. Even the most common matrix operations are realized in this class. So normal operators are overloaded to achieve this. E.g. transposing, inverting and multiplying matrices is now very easy to integrate in C++ code.

GRAPHICAL ENGINE IN C/C++

The graphical engine is taken from a tutorial [DIGI]. It is based on OpenGL. The keyboard and mouse input is handled by SDL [LIBS].

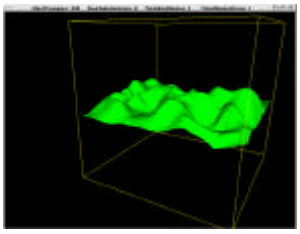

**Figure 2:** Octree - root node

It supports a camera class with which one can navigate in the scene with the help of the arrow keys and the mouse.
The speed of the movement is adjustable in the camera.cpp file.

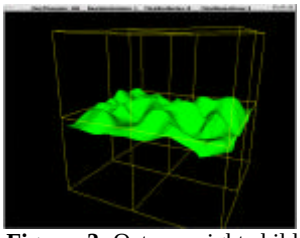

**Figure 3:** Octree - eight child nodes

The Init.cpp file and the Main.cpp file set up the OpenGL window, show the sky box and the bottom, and control the keyboard functionality. The Octree.cpp file is a class which stores the terrain file. If the terrain file is rather large, an intelligent way of storing the data is necessary. Frustum.cpp is a file which calculates the actual viewport in order that only visible parts are rendered. This is again for performance purposes.

For a better understanding, the purpose and the functionality of the octree is described in a little more detail. A scene consists of different objects. These objects are represented through a number of vertices which are connected either to lines, triangles or to quads. With these simple geometries, all higher level geometries can be described. A terrain can, therefore, consist of many triangles which all together make the impression of a surface. Each single triangle has to be rendered and this is a time consuming task. In order to speed up this process, the vertices need to be stored in an intelligent way.

For two dimensional problems, binary trees or b-trees exist. For the three dimensional problem, it would be best to use octrees. An octree is built up in the following way. First, the cubic space (cp. fig. 2) in which the terrain lies is divided into eight equal cubes (cp. fig. 3). Now every cube is again divided into eight equal cubes. This is done until a certain recursive depth is reached.

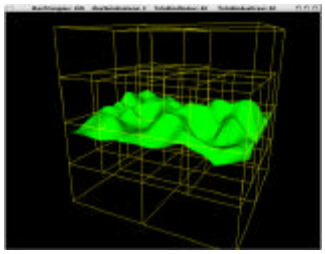

**Figure 4:** Octree - end nodes

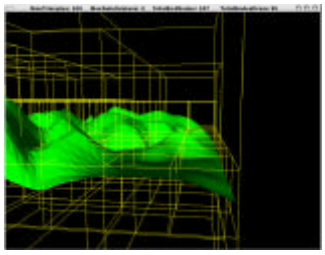

**Figure 5:** Octree - end nodes

Now a tree is built up with one root node, which has eight child nodes. Each of these eight child nodes itself has eight child nodes and so on. The end nodes, the nodes that do not have a child node, store the vertices of the terrain file which lie within them. Child nodes that do not contain vertices are abandoned. This can be seen in figure 4 and figure 5. If only a part of the scene is shown, few end nodes are situated in the displayed region. Then only the vertices which are stored in these end nodes are rendered. This is an enor Chapter mous performance boost, because not the whole scene is rendered in every picture frame. But the visualization is not the only advantage of an octree. Also object interference can be calculated in less time. For instance, the collision between a robot leg and the terrain needs to be found to obtain the [x, y, z]T vector. If the terrain was not stored in an octree, one would have to check every triangle whether it collides with the robot leg or not. If the terrain is rather detailed, this would mean several hundred collision checks every picture frame. With the octree it is first checked, to see if the robot lies within the root node. If this check is true, then each o the eight child nodes is checked. This is done until the end node in which now only a few triangles remain is reached. Instead of several hundred robot foot - triangle collision checks, maybe only ten robot foot -cube collision checks need to be done and then some few robot foot - triangle collision checks still need to be calculated. This is again an enormous speed up.

GAITS

Legged creatures have different gaits. Depending on their walking speed, they change these gaits accordingly. Some research has already been done for four and six-legged creatures. Their gaits are described in detail in [STIL00]. Six-legged creatures use three different speed levels for walking straight forward. These different speed levels result in three different gaits, which are

1. the tripod gait
2. the ripple gait
3. the wave gait

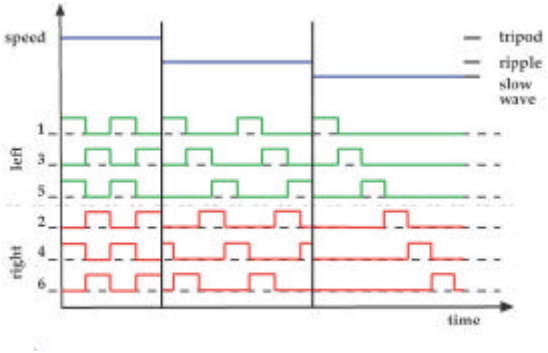


**Figure 6:** Phase and speed of gaits

The tripod gait is the fastest gait in which three legs have ground contact at the same timeand the other three legs are lifted and swung forwards. Figure 6 illustrates the phase and speed of the gaits. The direction of movement is given. The legs are numbered in the same way they are in C code.

DESIGN OF A BIO INSPIRED ROBOT

The aim of this project is to design a biologic inspired walking robot. The orientation on nature was successfully implemented in many kinds of developments in the last years. Especially in the field of robots nature has been taken in many projects as an example.

**Figure 7:** The whole hexapod

Often there is a big aim to rebuild the whole nature. This is very difficult and chanceless of being successful because the head start of nature is simply too big. To make reverse engineering and try to rebuild nature in every detail is failed to doom. The project should be orientated on principles found in nature by reviewing the technical possibilities they are able to achieve. And as a conclusion of those possibilities they should take the furthest development as possible. The concept for the design of the robot was orientated on the following needs.

- **ability to walk in rough terrain**
- **Bio Inspired**
- **reacting compliant on surface contact with an adjustability**
- **adjustment of contact area**
- **protection against dusty environment rough terrain**

To fulfill the target to walk in rough terrain a kinematic structure is required which allows to:

- climb obstacles which have the height of the body
- walk into holes which have the depth of the body height
- and to be able to turn around in every position

The kinematic structure defines the number of d.o.f and their arrangement in which they are implemented. This question can just be answered seriously in a full workspace and kinematic analysis is done. This is one of the concurrent engineering topics in this work. There is done a small joint motions analysis which had to be verified by the simulator on demand.

BIOLOGICALLY INSPIRED

The term Biologically Inspired and biologically orientated means to analyze especially the relevant animals. To focus the view on the the whole field, Anthropoids had been taken into account. Then the species were categorized by dividing into advantages and disadvantages with the following criteria:

- walk in rough terrain
- climb obstacles in body height
- turn at one position
- stable in every motion periods

On those species the project was orientated to discover the:

- kinematic structure
- dimensions
- joint behavior
- motion behavior and movements

of those animals to translate them into this robot project.

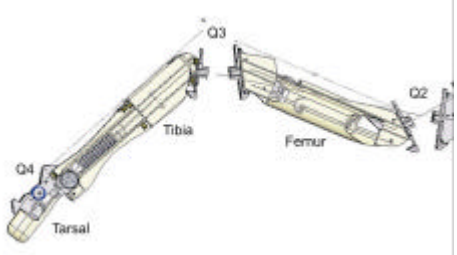


**Figure 8:** The front Leg

The results of this analysis are developed in percentages of dimentions of the whole animal. Figure 8 shows the leg which is designed in this project.

COMPLIANT TO THE ENVIRONEMENT

Reacting compliant to the environment is one goal in almost every robotic application whenever contacts occur between the robot and the environment. There is all the time a trade off between compliance and stiffness to avoid a robotic structure that is not adaptive enough for the environment or to sloppy in free motions.

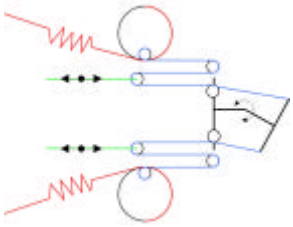

**Figure 9:** Routing of the compliance

**Figure 10:** Non Linear Pulley

If the trade off is not performed carefully, a noncontrollable mechanism could be the result. The best would be to implement an adjustable compliance over a large range. Therefore a comparison of different compliance solutions is done. The compliance which is implemented (see figure 9 ) is orientated on the AMASC, a project form Jonathan W. Hurst, Alfred A.[Jona04].

**Figure 11:** The Femur Top

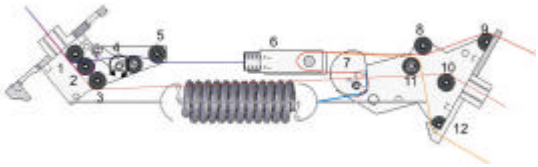


**Figure 12:** The femur cable routing

This limb includes the whole compliance for the following Q3 joint. Picture 11 shows the top view on the limb. Just right of the left joint Q2 there starts the gliding channel for pulley forks. Those gliding channels are covered with a light and thin (o.5 mm) sinter bronze metal layer to reduce friction. In them the both forks for the pulleys which realize a rotation of the joint are sliding. Just right of them the eccentric pulley is placed.

The right and left cable from top view ( see picture 12 dark blue) are for activating the joint Q3. They are passing the two pulleys 1 after they go over the compliance adjustment 4, over the pulleys 5 to the two forks 6. The compliance adjustment is a simple mechanic lever system which is activated via the violet cable which comes over pulley 2 and tears the lever from 4 in direction of the arrows. With this motion the two pulleys of the element 4 are moving on a circle up and elongate like this symmetrically the both dark blue cables for the activation of joint Q3.

This rotation is limited on 90 in the lever 4 which is enough to achieve in the eccentric pulley 7 ( see figure 10 ) a rotation over almost 180 to adjust like this the compliance. The same holds for the orange cable from the bottom of the joint Q3, which just comes over the pulley 12 to pass the position sensor on pulley 11 and then also goes to the forks to be afterwards fixated on the other one of the eccentric pulleys. The eccentric pulley 7 themselves are again fixed via a cable to the springs over the big radius.

ADJUSTMANT OF CONTACT AREA

If the robot walks in rough terrain there can be many changes of the consistency of the ground material. There can be for instance a rocky surface which is very stiff in the contact point or a muddy or sandy surface which has totally other physical attributes.

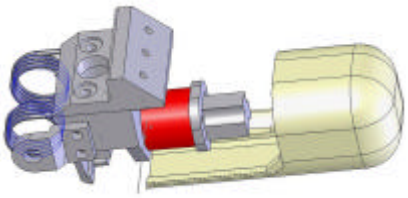

**Figure 13:** Tarsal Limb

To be able to adapt to those changes of environment without changing the end-effector, the contact face of the end-effector needs to change in order to have efficient traction. In insects this is done by using their last leg limb the Tarsal. Because of this, adjustment of contact area is implemented in this project by using the last limb of the leg.

PROTECTION AGAINST DUSTY ENVIRONEMENT

To protect the mechanism from dusty environment, means to protect sensitive devices such as ball bearings. This could mean not to use conventional bearing systems because they are reacting on temperature and dirt minimum with decreased lifetime. The joint design and achievable performance was one of the big defiance of this work. To reach this goal the robot had to get different solutions for every joint according to the needs of this particular joint. Considering the jobs a joint has to fulfill these are the 3 major functions:

1. guidance
2. activation
3. compliance

The guidance is discussed deeper in the thesis, activation also and the compliance is explained above. Those three different features were addressed separately for each limb.

The *joint Q2* is realized with the *crossed blade spring* and the *additional shanks.*

This decision has been taken due to the advantage that this joint will work

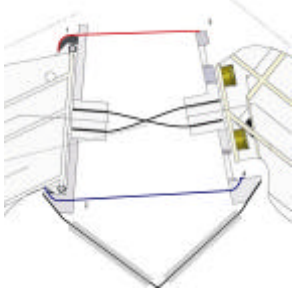

**Figure 14:** Joint Q2

fine in this project without using conventional bearings.

This kind of guarding the joint was proven in the tests as the best way to guard without rotatory devices. In this setting one has the lowest moving of the center of rotation as well as the most stiffness against cross forces. The only problem of this joint setup is that the blade springs maybe be banded out of the elastic field into the plastic deformation.

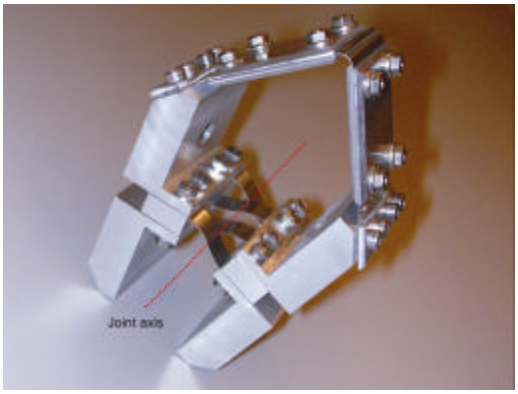


**Figure 15:** Test example Joint Q2

This is not a problem in the application directly but is going to tire the material out in terms of lifetime of the joint. When considering these facts it was decided during the research period to use only spring steel because of its much longer lifetime and not polymer structures which are much more complicated from the point of view of analyzing the motion behaviour.

RESUME

This paper allows a short view about in Thises the whole paper discussed the theories in more detail. Discussed in here are the realisation of hexapod simulator in C. Different solutions for the use of open GL in context with C and C++. Further on are suggestions for different design solutions.
The design solutions are mostly orientated on the questions of Biologically inspired , compliance behaviour and on joint solution which are protected against dusty and unfriendly environment.

BIBLIOGRAPHY

[DIGI] DIGIBEN: GameTutorials - Game programming with a personality.
http://www.gametutorials.com (Sep 07,2004).

[JONA04] JONATHAN W. HURST, Alfred A. R.: An Actuator with Mechanically Adjustable
Serial Compliance. (2004).

[TECH] TECHSOFT: Matrix C++ template class library with full source code.
http://www.techsoftpl.com/matrix/ (Sep 07,2004).

[LIBS] LIBSDL.ORG: Simple DirectMedia Layer. http://www.libsdl.org (Sep 07,2004).

[STIL00] STILL, Susanne: Walking Gait Control for Four-Legged Robots, University of
Hannover, Diss., 2000.