Development of a Flight Control Algorithm for the Autonomously Flying Robot MARVIN

M. Musial U. W. Brandenburg G. Hommel
Technische Universität Berlin
Department of Computer Science – Real-Time Systems & Robotics Group
Contact: *http://pdv.cs.tu-berlin.de/MARVIN/*

Abstract— **MARVIN is an autonomously flying robot based on a model helicopter. It has been designed for participation in the International Aerial Robotics Competition (*IARC*) Millennial Event 1998 – 2000. The competition task consists of a search operation in an unknown environment with possible threats such as fires, water fountains, and smoke. For autonomous flight, a flight control algorithm runs on the on-board computer that stabilizes the helicopter and enables it to reach a way-point issued by the ground station. The flight control algorithm consists of a hierarchy of modified PID controllers. By MARVIN's performance, it has been proven that no more sophisticated approach to control is required for helicopter flight control.**

I. Introduction

MARVIN is an abbreviation for **M**ulti-purpose **A**erial **R**obot **V**ehicle with **I**ntelligent **N**avigation. MARVIN is an autonomously flying robot that can fulfill a complicated search mission purely on the basis of sensor data, without any human interaction.

The requirements to be fulfilled by MARVIN are defined by the mission of the International Aerial Robotics Competition Millennial Event, as described in [1]. The mission task consists of finding and classifying hazards and victims in a simulated disaster area. There are black drums containing dangerous materials, with the contents distinguishable by symbols on the drums' surfaces, and there are fires, water fountains, and smoke threatening the operation of the robot. Victims may be dead or injured persons, the survivors recognizable by motion and sound. In the 2000 competition finale, MARVIN was the only participant to recognize target objects during autonomous flight. Thus, it won the Millennial Event.

This paper deals with the approach to and the development of the flight control algorithm for MARVIN. This flight control algorithm is responsible of flight stabilization of the inherently unstable air vehicle and of reaching a single target way-point issued by the ground station. This means that mission strategy is not part of "flight control" and consequently not covered here. Please refer to [2], [3] for details on the higher levels of control. The following section of this paper gives a short introduction to the overall system. Then, the approach to flight control is explained. After some details about the implementation and the testing process, the performance of MARVIN's flight control is described.

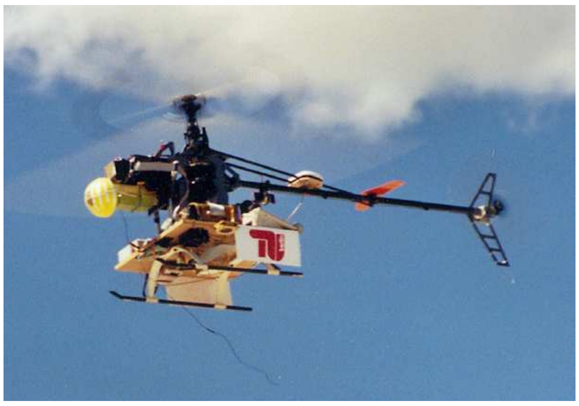
Fig. 1. MARVIN in flight

II. System Design

Figure 2 provides an overview of the MARVIN system. During system development, every detail has been carefully optimized with respect to weight, prize, power consumption, and capability. Much care has been taken to select components that fulfill the respective tasks and integrate smoothly into the system, always avoiding things being "better than needed" at the expense of some of the criteria mentioned above. Basically this made MARVIN's success possible.

A. Air Vehicle

The basis of the MARVIN air vehicle is a conventional model helicopter. It has a rotor diameter of 1.9 m and is equipped with a two-stroke chain saw petrol engine producing about 2 kW. Its takeoff weight amounts to about 11 kg, operation time is approximately 30 minutes. Figure 1 shows a view of MARVIN in flight.

B. Sensors

The sensors on board MARVIN are:

- A hand-crafted inertial measurement unit consisting of three magnetic field sensors, three semiconductor acceleration sensors, and three piezo-electric rotation sensors (material cost below $250).
- A flame sensor that detects certain ultraviolet light characteristic of burning wood, gas, or oil.
- A light barrier RPM sensor for measuring the main rotor RPM.

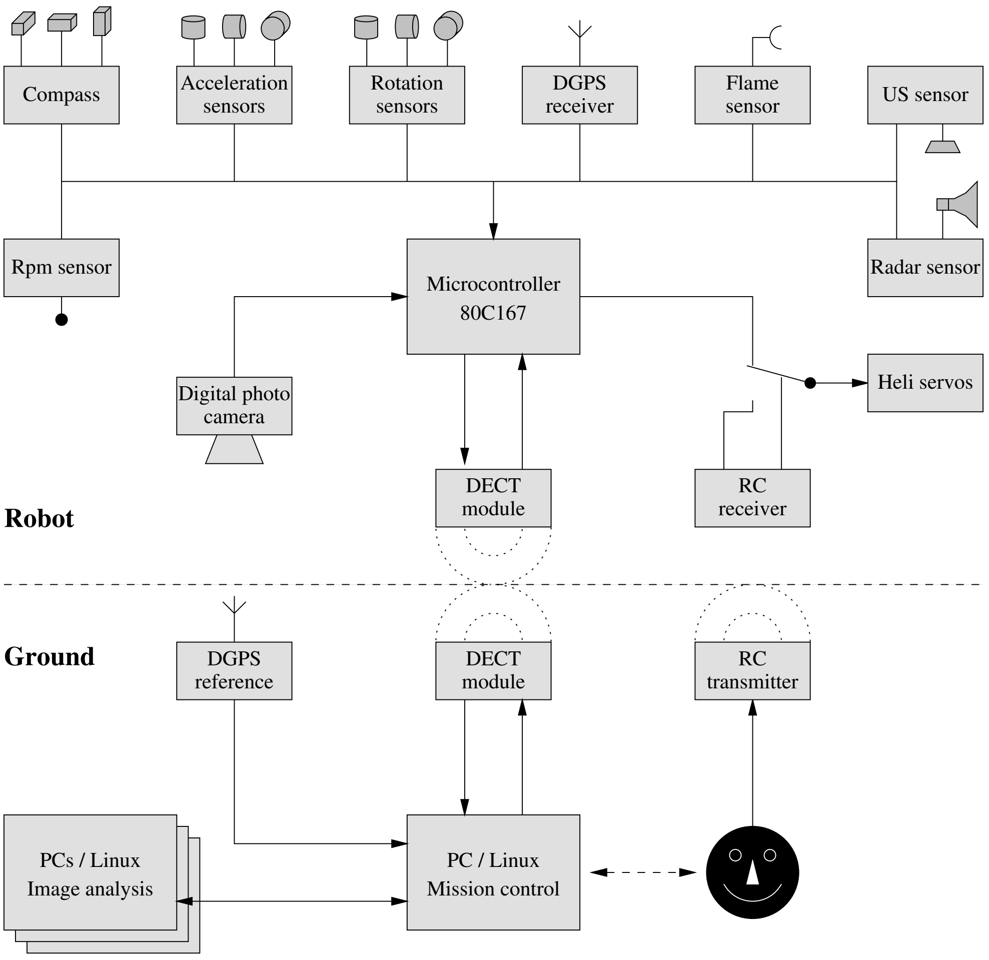

Fig. 2. Overview of the MARVIN system

• An ultrasonic echo sensor looking down.
• A radar sensor, based on frequency modulation and frequency shift detection, looking ahead.
• A digital photo camera looking down.
• A NovAtel RT-2 carrier phase differential GPS receiver (thanks to NovAtel).

C. Computers

The only source of computational power on-board is a single-chip computer, an Infineon (formerly Siemens) SAB80C167 microcontroller. This device is perfectly capable of interfacing all the on-board peripherals without additional circuitry. In addition to interfacing the peripherals, the C167 runs the flight control software described in this paper.

A network of Laptop PCs under Linux serves as a ground station for image processing and the "mission control" software.

D. Communication

Data communication between MARVIN and the ground station is performed via two pairs of Siemens Gigaset M101 Data communication modules. Each pair provides a wireless serial "null-modem cable" connection at 107 kbit/s full-duplex, yielding 214 kbit/s of bandwidth.

The communication software implements a *shared memory* approach to communication: Both the on-board computer and the ground station modules use a dedicated memory area containing *all* the variables relevant to the system state. Whenever data are changed within the shared memory area either on-board or on ground, this change is distributed to all remaining copies of the shared memory through the exchange of data packets.

III. Approach to Flight Control

The flight control – or autopilot – module first has to provide flight stabilization, since a helicopter is by itself an instable air vehicle. Second, it has to follow a linear flight path segment determined by "mission control" at the ground station.

The autopilot uses a hierarchy of controllers with bounded piecewise linear transfer functions. All of the controllers get one state variable x and its derivation $\dot{x}$ as inputs and compute an answer y as follows:

$$\begin{aligned} y_i &:= y_i + f_3(x) + f_4(\dot{x}) \\ y &:= y_i + f_1(x) + f_2(\dot{x}) \end{aligned}$$

That is, y_i serves as a memory for controller outputs being integrated over time. These outputs are computed according to the transfer functions f_3 and f_4, whereas f_1 and f_2 affect the answer y directly. Thus

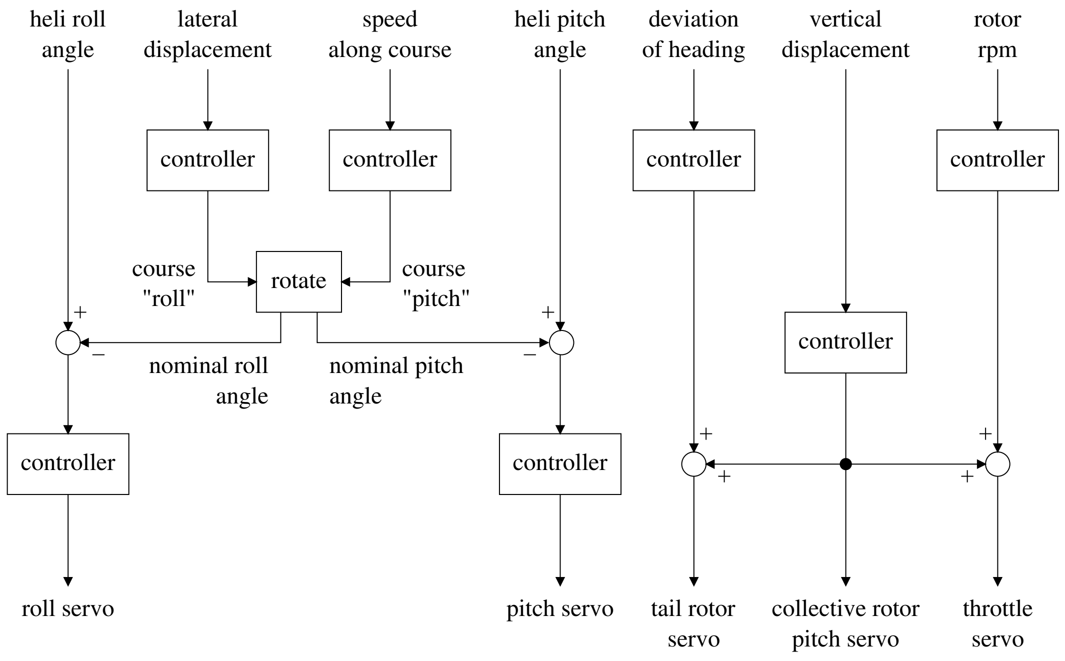

Fig. 3. Architecture of the MARVIN flight controller

there are P (proportional), I (integral) and D (differential) components in the controllers. Since the transfer functions are piecewise linear with few support points, the elementary controllers are just slight generalizations of a classical PID controller. The somewhat unorthodox manner integration is performed in has been chosen because it allows to force the current controller output to a certain value by just calculating the required value of the integrator variable y_i. This enables the microcontroller to softly take over control of the helicopter: The microcontroller permanently measures the servo signals as long as a human pilot is steering via remote control. When the pilot software is instructed to take over control, the controller outputs are initially forced to the very same values the human pilot has used. While this is not required for autonomous takeoff, it constitutes an unconditional prerequisite for in-flight takeover during the development phase.

Figure 3 depicts the architecture of the flight controller. Each “controller” rectangle represents one of the elementary controllers as described above. Control of the cyclic rotor pitch setting is performed in two stages: MARVIN’s speed and position relative to the course line are used to calculate designed orientation angles, because the helicopter has to be tilted in order to move in a certain direction. These designed orientation angles are rotated into the helicopter’s current heading angle and then compared with the current pitch and roll angles, yielding the steering signals for the cyclic rotor pitch setting.

IV. Implementation and Testing of Flight Control

The flight control software is implemented in C and runs 20 control cycles per second. Thanks to the implementation in C and the shared memory communication architecture, the flight control software can be run on a PC under Linux without any change to the controller sources. For testing purposes, a very primitive flight simulator program has been written that provides interface functionality and simply includes the original controller sources. In this simulation, the controller gets certain target coordinates at predetermined times. The result of the simulation is a log of the state and output variables that can be easily plotted and evaluated. Thus, every change to the controller software or the controller parameters can be checked off-line for basic reasonability. Figure 4 shows an example of such a simulation plot. The way-point sequence here was: Takeoff and climb to 20 m at t=30 s, move west (positive y) by 20 m at t=80 s, descend to 5 m at t=130 s, move north (positive x) by 20 m at t=180 s, climb to 20 m again at t=230 s, land at t=280 s. The position-over-time diagram is at the upper right of the figure, projections of the trajectories can be found below. In the left column, the speed, rotation rate and rotation angle functions are depicted from top to bottom. In the bottom right corner, the rotor RPM (`Wr`) and the servo outputs are shown.

Because of the simplicity of the simulation software, it has to be noted that a successful flight in the simulation is a necessary, but not at all sufficient condition for success in controlling the real vehicle. For the in-flight testing of the flight control software, the helicopter is started under manual control by a very experienced model helicopter pilot. While hovering at an altitude of at least 20 m above ground, servo control is switched to the on-board computer. The human pilot can gain back control at any time by throwing a single switch on the remote control unit, and from 20 m, he has plenty of time to recover the vehicle safely, no matter what the control software was doing. This approach to testing the controller has proven re-

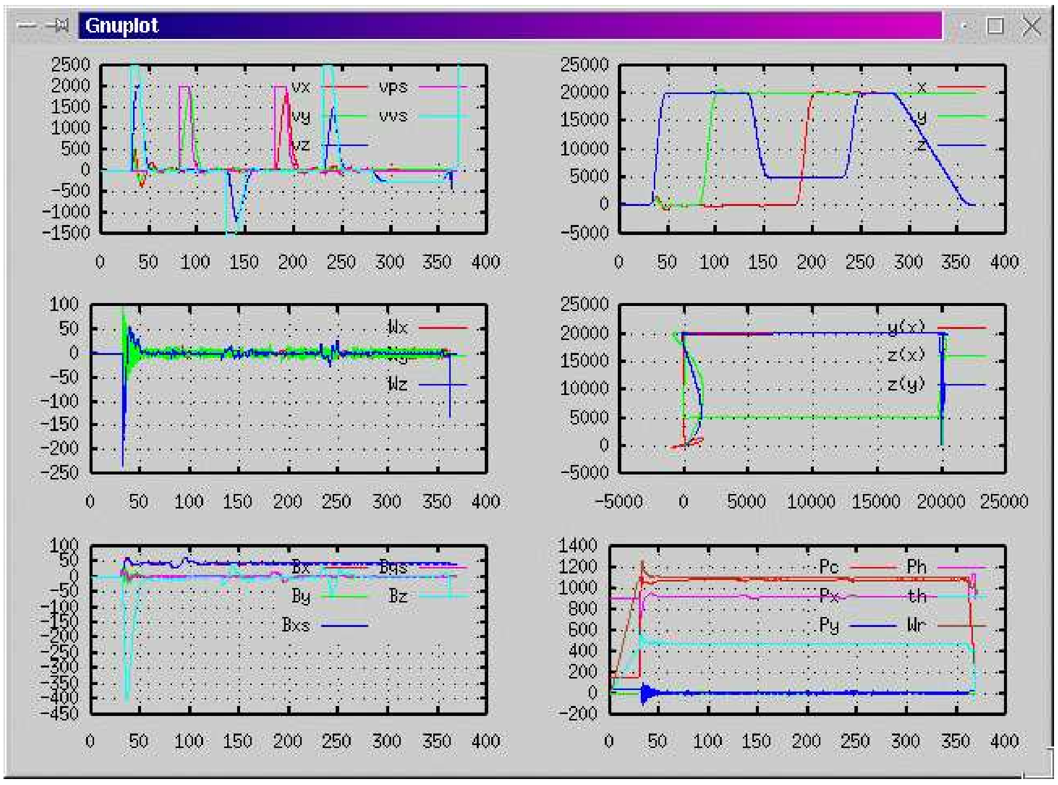

Fig. 4. Example plot of a control simulation

liable and successful: The first in-flight take-over was performed on May 23rd, 1999, only one month before the first successful demonstration of MARVIN in the 1999 qualifier competition.

Since helicopter flight dynamics are complex, dependent on many parameters and composed of a huge number of different effects many of which are of little practical influence on the controller, the controller parameters have not been derived from a mathematical model of helicopter dynamics. Instead, after rough checking against the simple simulation mentioned above, the parameters have been adjusted by means of in-flight experiments. In order to alleviate and fasten this process, the controller parameters are part of the shared memory area of the communication system. Thus, the parameters can be changed from the ground station at any time and transmitted to the helicopter even when it is still airborne (it has to be under *manual* control, however). For easy access to the parameters, they are determined in a single configuration file. After a change to the parameters, the configuration file is parsed by a small utility program that transmits the parameter setting to the helicopter via the shared memory communication.

Figure 5 depicts a small excerpt from the parameter file, defining the transfer functions for the nominal pitch and roll angle controllers in hovering mode, and the resulting graphs of the transfer functions. One can see that the parameter file is in rather symbolic notation. The symbol Bqs defines some bounds on the controller's output. The next two lines are the support point coordinates for the positive and differential input, respectively, that is, the position error and speed, in mm resp. mm/s. The following four lines determine the respective output values $f_1(x)...f_4(\dot{x})$ in 0.1° resp. 0.1°/s. The *- stands for continuation with symmetry to the origin. The last two lines defining Bqs_pq and Bps_pp compose the support point matrices for the two – identical – controllers. In the graphs, it is visible that outside the support point intervals, the transfer functions are constant in order to limit the controller output in the case of extreme errors.

V. Results

The relatively simple controller architecture described above can successfully stabilize and control the robot. In hovering above a fixed target position, the flight control software performs much better than a human pilot, who always suffers substantial difficulty in estimating the helicopter's movements along the view direction.

However, since the controller is more or less a PID controller (exactly a PID controller in a certain region around the target location), it suffers from the inevitable tradeoff between fast convergence and the tendency of oscillation. In rare cases self-amplifying oscillations have occurred. It is possible, but not yet confirmed, that errors in the readings of the electronic compass are primarily responsible for these unstabilities.

Figure 6 depicts the nominal and actual positions (z is altitude) during a typical autonomous mission flight. Note that the delayed compensation of the y-error after $t = 52980$ is intentional, the controller is waiting here for the helicopter to change its heading.

```
# Controller Bqs, Bps: pointmode
Bqs     =     -200    200    -250 250    12

i_pq    =  -1000   -200           *-
i_pq1   =  -5000  -1000           *-

o_p     =     80     50           *-
o_p1    =    100     60           *-
o_i     =     12      4           *-
o_i1    =     15      5           *-

Bqs_pq = i_pq o_p i_pq1 o_p1 i_pq o_i i_pq1 o_i1
Bps_pp = Bqs_pq
```

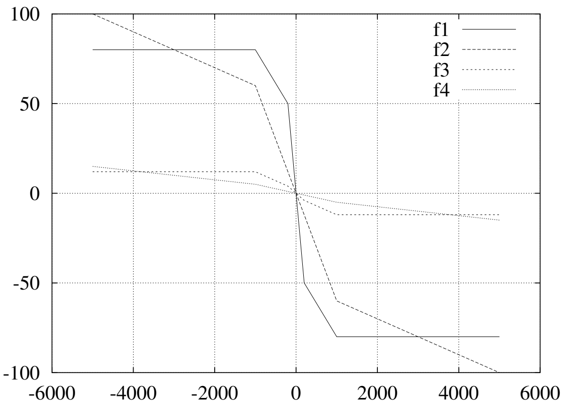

Fig. 5. Excerpt from the parameter file and the stated transfer functions f_1 to f_4

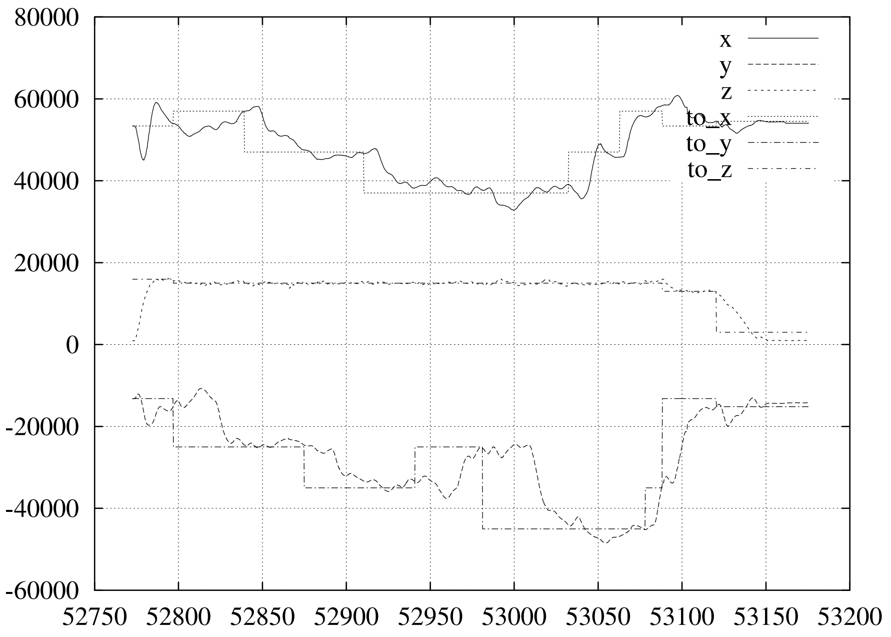

Fig. 6. Nominal and actual positions during a flight

VI. Conclusion

In this paper, an approach to and the implementation of a flight control algorithm for a model helicopter has been described. The resulting controller has proven successful, showing that a quite difficult control task can be solved easily by a rather conventional approach and without an exact mathematical model of the dynamics of the system.

At the time of writing, an alternative approach to the flight control of MARVIN has begun to be investigated, based on trajectory planning. This novel approach will, if it turns out to be successful, greatly decrease the number of parameters needed, assign more physical meaning to the remaining parameters and largely solve the problem of oscillations.

References

[1] International Aerial Robotics Competition: The Robotics Competition of the Millennium. http://avdil.gtri.gatech.edu/AUVS/CurrentIARC/IARC2000Intro.html.

[2] M. Musial, U. W. Brandenburg, G. Hommel. MARVIN – Technische Universität Berlin's flying robot for the IARC Millennial Event. In *Proc. AUVS Symposium*, Orlando, Florida, USA, 2000.

[3] Marek Musial, Uwe Wolfgang Brandenburg, Günter Hommel. Cooperative Autonomous Mission Planning and Execution for the Flying Robot MARVIN. In Enrico Pagallo, Frans Groen, Tamio Arai, et al., editors, *Intelligent Autonomous Systems 6*, Amsterdam, Berlin, Oxford, et al., 2000. IOS Press and Ohmsha.