Navigation and Control Software Design and Implementation for Lunar Rover Validating System

Chunguang Bu
State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences
Shenyang 110016, China
cbu@sia.cn

Zhenwei Wu and Xiaokang Song
Graduate School of the Chinese Academy of Sciences
Beijing 100039, China
{wuzw, songxiaokang}@sia.cn

Abstract – This paper provides the design and implementation of navigation and control software for an aerospace-level lunar rover validation system. Based on analysing structure characteristics of the navigation and control system, real-time navigation and control software involving interrupt service routine, tasks, and task-task communication is designed and implemented within the embedded operation system VxWorks. Although the current research of our project is still on the way, this navigation and control software designed here is already successfully applied in some of physical experiments on the lunar rover platform. The practical employment indicates the utility and reliability of the designed software totally satisfying design requirements of the navigation and control system.

Index Terms - Navigation and control, Real time multi-task, Lunar Rover, Software Design

I. INTRODUCTION

As the sole natural satellite of Earth, the moon has been deemed to be a preferred space base and transfer station to outer space. Researches have found that the moon has large reserves of helium-3, which can be used over 7000 years if developed as a nuclear material to generate electricity for human beings. Besides its great potential of mitigating the current status of energy resources shortage on Earth, the moon also is an ideal space base for scientific experiments and astronomic observation. With the consideration of the above reasons, the growing international interest in lunar surface exploration has been triggered off. In order to perform lunar surface exploration missions, different lunar rover platforms carrying manifold application payloads have been designed to semi-autonomously or in a remote way traverse and take scientific samples on the lunar surface. As one of the key technologies for lunar rovers, autonomous navigation and control technology involve several foundationally theoretical problems including modeling, perception, navigation and control. The validation of algorithm research should be linked with engineering demands.

In order to provide an auxiliary way to test all the developed algorithms of navigation and control, a lunar rover validation system for ground demonstration has been built by the Shenyang institute of Automation (SIA), the Chinese Academy of Sciences (CAS). The aerospace-level embedded power computer and real-time multi-task operating system VxWorks have been used as the hardware and software platform of navigation and control for the lunar rover validation system, which provides a convincing testing environment for lunar rover’s navigation and control algorithms. According to project arrangement, the research group in SIA designed the main control system software structure, and developed the corresponding navigation and control software for the lunar rover.

II. BRIEF DESCRIPTION OF LUNAR ROVER VALIDATING SYSTEM

A. Lunar Rover Validating System

The lunar rover Validating System is shown in Fig. 1. The whole lunar rover validation system involves two main parts of a simulative ground monitoring and control station and a lunar rover. Two communication channels including the measuring and controlling channel and the data transmission channel work between the simulative ground monitoring and command station and the lunar rover. The measuring and controlling channel with low bandwidth works bidirectionally for transmitting quasi-real-time data such as the remote control commands and running status of the lunar rover. The data transmission channel with higher bandwidth than the measuring and controlling channel is mainly responsible for sending large but not real-time data to the simulative ground monitoring and command station.

The Lunar rover is comprised of mobility platform, manipulator subsystem, mast subsystem and on-board integratedly electronic subsystem. The manipulator equipped with a polishing tool at the end is tele-operated through macro-commands from the monitoring and control station to polish object rocks autonomously. The mask subsystem equipped with navigation cameras is controlled through macro-commands from the monitoring and control station to turn the navigation cameras toward the expected direction. The mobility platform is composed of six-independently driven wheels and four-independently steering joints and two-independently extending joints. By navigation and control software, lunar rover can perform Autonomous navigation.

The on-board integrated electronic subsystem includes the hardware and software for the rover’s navigation and
control, data management and payload system. Two computers named as main control computer and payload computer are mounted on the lunar rover. The two on-board computers can communicate with each other in two manners of 1394 field bus and computer PCI field bus. They work through 1394 field bus in normal conditions. If one of the on-board computers runs abnormally, its tasks of resource manage will be performed through PCI field bus by the other computer working normally.

The software running on the main control computer includes two parts of navigation and control program and data management program, while the payload computer comprises three parts of application payloads management, data memory and data transmission programs. Navigation camera, obstacle avoidance camera and stereo vision software are managed by the payload computer.

The navigation and control software receives information about obstacles through calling for the data management interface functions. Besides, the data management software is employed to transfer camera images data to ground monitoring and control station for the rover’s global path planning.

B. Vxworks Operation System

Vxworks is a Modern real-time systems which are based on the complementary concepts of multitasking and intertask
communications. A multitasking environment allows a real-time application to be constructed as a set of independent tasks, each with its own thread of execution and set of system resources. The intertask communication facilities allow these tasks to synchronize and communicate in order to coordinate their activity. In VxWorks, the intertask communication facilities range from fast semaphores to message queues and from pipes to network-transparent sockets[1].

Another key facility in real-time systems is hardware interrupt handling, because interrupts are the usual mechanism to inform a system of external events. To get the fastest possible response to interrupts, interrupt service routines (ISRs) in VxWorks run in a special context of their own, outside any task’s context.

III. SOFTWARE DESIGN REQUIREMENTS

As shown in Fig. 2, the architecture of the lunar rover navigation and control system is composed of three layers including task layer, behavior layer and action layer, developed according to hybrid system theory and behavior-based control theory[2]. The task layer is responsible for task planning, which means to perform local path planning and re-planning based on the system’s interior and outer state. The behavior layer is employed to generate the rover’s behaviors. The action layer refers to control hardware of motor servo systems and sensors. Explicit specification requirements for the expected navigation and control software are designated according to the task and behavior layers.

The functions of the main modules in task and behavior layer are summarized as follows.

- Remote commands interpreting
  Since the bandwidth of the measurement-control channel for the lunar rover is limited, the commands sent by ground measurement and control station must be transferred according to aerospace engineering standard protocol. Therefore, the command data sent by ground station have to be unwrapped by the main control software according to transfer protocol to obtain the proper data expression used by control system.
- Path planning
  Motion commands received by the lunar rover are macroinstructions of path point sequence. Each path point defines the expected location, azimuth angles and motion velocity of the rover. The main control software will analyze the path point sequence into specific motion behaviors, and monitor the implementation of these motion behaviors.
- Autonomous obstacle avoidance
  Autonomous obstacle avoidance means the process of re-planning local path if the rover runs into obstacles that the rover cannot get over. Whether implement autonomous obstacle avoidance or not depends on how rough the obstacles will be. The current implement period of this process on our lunar rover is four seconds because of the limit of on-board computer processing ability and the complexity of stereo vision based obstacle detection and recognition.
- Autonomous state estimation

Trajectory control of the lunar rover actually is to control the rover to a specific position with reference to landing vehicle. So it is necessary for the rover to be able to determine where it is. Three different sensors including wheel encoders, inertial navigation systems and wireless sensor network have been equipped with the lunar rover. Inertial navigation data are received via RS232 serial port, while wheel encoders and wireless sensor network are connected through CAN field bus.

- Trajectory planning
  The trajectory planning module is used to compute the trajectory target for next control step based on next waypoints achieved by the path planning module, the rover’s current position and attitude. Executive time of this module is 200 milliseconds.
- Vehicle inverse kinematics
  The trajectory object is solved to be revolution speeds of the driven wheels and joint positions of the steering wheels through the rover’s inverse kinematics. Several constraints involving terrain characteristics, wheel slips, rover body configuration and joint speed limits are taken into account. The control period of multi-joint motion is restricted to 200ms because of the limits of hard communication bandwidth.

IV. SOFTWARE DESIGN

The VxWorks based software design includes application software design and operating system software customization for specific target platform. Application software design includes a series of works involving design tasks classification, task priority determination, communication way selection, shared memory protection design.

A. Task Classification

Software is a implementation tool for applications. During designing the application software, both application functions and software platform itself should be integratedly taken into account as a whole. Task classification is an important part of multi-task software design methods. If task classification is of too much detail during design, the system will be burdened with increased expenditure by frequently employing task management module to process context tasks, while if task classification is of too much simplicity, the system may run a single task with too much time. Therefore, task classification is usually performed to the following principles.

1. Operations with independent but severely correlated function should be assigned as one task;
2. I/O operations should be assigned as independent tasks;
3. Operations involving large mathematical computations such as state estimation in this paper’s system should be treated as independent tasks;
4. Codes running periodically are usually treated as a task to execute, possessing independent stack space;
5. Operations with different task priority should be treated as different tasks; Tasks with highly real-time requirement have high task priority.

According to task classification rule and required functions of the navigation and control system, we design Lunar rover navigation and control software structure as...
shown in Fig. 3. Main interruption service routine, task is described as follows.

- **Software initialization**
  
  Software initialization task is used for creating and starting task, creating interrupt handler, claiming and creating semaphore and shared memory resources, setting initial value of global variables etc.

- **Clock interrupt handler**
  
  Generate trigger semaphores to tasks with different executing periods by frequency division of high-resolution system clock signal.

- **CAN Bus interrupt handler**
  
  Wheel encoder data and sensor network data are received through CAN bus in an interruptive way. The CAN bus interrupt handler is responsible for receiving the above data, putting them into a designated data shared memory area, and sending semaphore to specified tasks of data processing and filtering.

- **RS232 interrupt handler**

![Diagram](https://via.placeholder.com/150)

Fig. 3. Lunar rover navigation and control software structure
Inertial measurement sensor data are received through series port in an interruptive way. The RS232 interrupt handler stores the inertial sensor data at the designated data shared memory area, and sends semaphore to specified data processing and filtering tasks.

- Motor encoder data processing and filtering task
  Activated by semaphore sent by CAN interrupt handler, the encoder data processing and filtering task firstly reads original encoder data from the designated data shared memory area, and then produces joint angles and joint turning rates by executing the corresponding filtering algorithms.

- Sensor network data processing and filtering task
  Activated by semaphore sent by CAN interrupt handler, the sensor network data processing and filtering task firstly reads sensor network data from the designated data shared memory area, and then executes the corresponding filtering algorithms to produce the estimated position value of the rover.

- Inertial measurement sensor data processing and filtering task
  Activated by semaphore sent by RS 232 interrupt handler, the inertial measurement sensor data processing and filtering task firstly reads sensor raw data from the designated data shared memory area, and then executes the corresponding filtering algorithms to produce the motion state estimates for the lunar rover.

- Remote control command interpreting task
  The data management software receives remote commands of navigation and control for the lunar rover, and stores them at a designated shared memory area. Then The remote command interpreting task is triggered by message queues from the data management software to extract preprogrammed path point information out of remote control commands, and store the preprogrammed path point information at another shared memory area. After that, a flag variable is set to indicate that system clock task will send semaphore to real time trajectory control task to activate it.

- Path planning task
  Obstacle avoidance task activated by asynchronous triggering signals reads obstacle information from the designated sharing memory area, and re-plans a goal-oriented local path used as inputs to true trajectory control.

- Trajectory control task
  The trajectory control closed loop involves several control algorithms including rocker-bogie motion parameters estimation, multi-sensor based motion state estimation as well as trajectory planning and vehicle inverse kinematics computing, with large computation cost and strong interaction between control algorithms. With a view to system real time, all the above control algorithms are integrated into the trajectory control task. It is activated by periodically semaphore with the triggering period of 200 milliseconds.

- data logging task
  Activated by periodically semaphore sent by clock interrupt service routine, this task reads navigation and control related data from data-sharing memory area, which can be transferred to ground measurement and control station by using the data management software. The period of task execution is one second.

- Task exception handler
  Computer systems on lunar rover can be reseted by remote command from ground monitoring and command station, but for improving error tolerance ability, we designed Task exception handler. We can look upon it as software interrupt handler, when there is a error which cause this task suspended, Vxworks will trigger a singal, after receiving singal, task exception handler will do someing to restore software to a initial determinate state.

B. Intertask Communication

Intertask Communication is important way for multi-task coordination, VxWorks supplies a rich set of intertask communication mechanisms, including:

- Shared memory, for simple sharing of data.
- Semaphores, for basic mutual exclusion and synchronization.
- Mutexes and condition variables for mutual exclusion and synchronization using POSIX interfaces.
- Message queues and pipes, for intertask message passing within a CPU.
- Sockets and remote procedure calls, for network-transparent intertask communication.
- Signals, for exception handling.

The optional products VxMP and VxFusion provide for intertask communication between multiple CPUs.

The software designed in this paper is required to run in an environment with a single CPU and no network. Therefore, the ways of VxMP, VxFusion, Sockets and remote procedure calls are inappropriate for this paper’s application. The Message queues and pipes way is the standard way to synchronously communicate between each task, however it is probable to result in system communication obstruction if any design mistakes happen. This paper adopts a special communication mechanism of semaphore with shared memory and mutexes to improve software robustness. During the software design in this paper, semaphores are used as a signal to activate tasks, shared memory is used to store shared data, and mutexes and condition variables are used to coordinate read-write operations to shared memory data, with the consideration of data integrity.

C. Task Scheduling Algorithm and Task Priority

In Vxworks, default Scheduling algorithm is priority-based preemptive scheduling. You can also select to use round-robin scheduling for your applications, modules of navigation and control system presented in paper is different from each other about real time requirement, so in this paper, software use priority-based preemptive scheduling[3][4].

Task priority determination mainly depends on its real time requirement. Based on analyzing real time requirement of the above tasks, task priority is determined as in Table I. Smaller number corresponds to higher task priority. Because the triggering signals for encoder data processing and filtering
table and sensor network data processing and filtering task are both generated by CAN Bus interrupt handler, it is impossible for them to happen simultaneously. So the encoder data processing and filtering task and sensor network data processing and filtering task share the same task priority.

### TABLE I

<table>
<thead>
<tr>
<th>Task</th>
<th>Task Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory control task</td>
<td>15</td>
</tr>
<tr>
<td>Path planning task</td>
<td>17</td>
</tr>
<tr>
<td>Inertial measurement sensor data processing and filtering task</td>
<td>19</td>
</tr>
<tr>
<td>Encoder data processing and filtering task</td>
<td>21</td>
</tr>
<tr>
<td>Sensor network data processing and filtering task</td>
<td>21</td>
</tr>
<tr>
<td>Remote command interpreting task</td>
<td>23</td>
</tr>
<tr>
<td>Navigation and control data logging task</td>
<td>25</td>
</tr>
</tbody>
</table>

#### D. Task exception handler

Errors in program code or data can cause hardware exception conditions such as illegal instructions, bus or address errors, divide by zero, and so forth. The VxWorks exception handling package takes care of all such exceptions. The default exception handler suspends the task that caused the exception, and saves the state of the task at the point of the exception. The kernel and other tasks continue uninterrupted. Tasks can also attach their own handlers for certain hardware exceptions through the signal facility. If a task has supplied a signal handler for an exception, the default exception handling described above is not performed. In this paper, a task exception handler is designed for recovering software system from catastrophic events. `setjmp()` is called to define the point in the software initialization task where control will be restored, and `longjmp()` is called in the signal handler to restore that context.

However, when an ISR causes such an exception, there is no safe recourse for the system to handle the exception. The ISR has no context that can be suspended. Instead, VxWorks stores the description of the exception in a special location in low memory and executes a system restart.

#### V. CONCLUSIONS

This paper described the software design and implementation for navigation and control of an aerospace-level lunar rover based on analyzing structure characteristics. Interrupt service routine, task classification and intertask communication are designed and implemented according to multi-task programming of VxWorks except Sensor network data processing and filtering task,program code is written by C language. The whole software design and implementation is performed based on modular ideal and multi-task operating system. The developed software for navigation and control has been successfully utilized in the ground demonstration experiments and navigation and control algorithm testing experiments of lunar rover. CPU starvation is tested by WindView which is a VxWorks tool component, CPU starvation is about 50%, is good status for CPU.

#### REFERENCES