Building a Shelter on Mars

Jesika H Haria
6.141J Robotics Science and Systems I
Challenge Design Exercise

March 11, 2013

1 Problem Statement

The goal of this exercise is to design an effective strategy for an autonomous robot to build a shelter on Mars. Using sophisticated perception, navigation and manipulation skills, the robot will be able to navigate through a noisy environment with partial map information, find and store the building blocks and finally assemble them into a primitive shelter. The success of this design will be rooted in judicious selection and arrangement of hardware, the robustness of the code guiding the robot's decisions and the reliability of the interface's communication channels.

2 Assumptions

Some assumptions regarding the nature of the challenge, environment and robot need to be made in order to design the specifications of the system.

Environment It is assumed that the environment is bounded within reasonable distance, flat, and smooth; all three qualities are essential for the physical safety of the robot. The entire space that robot traverses will be around $10m^2$, give or take $5m^2$. The blocks are expected to be cubic or rectangular in solid colors of either red, blue or green, with each side measuring at least $5cm$ and at most $10cm$, a single block weighing less than $500g$, to ensure reliable pickup, on-board storage and drop-off. There should be around

20 blocks.
It is also assumed that the robot has a previous, although partial, but useful map of the world, with the ability to update its map. The assumption is that at least 50% of the blocks are in their known locations and sizes.

Challenge The challenge will involve the robot to pick up at least 50% of the twenty building blocks in the map and assemble it into the shelter in the form of a wall within the time limit of 15 minutes. Note that this is a loose estimate, bound by the assumption that at least 50% of the blocks are in their known positions; we do not know whether the other 50% even exist. During the period of the challenge itself, the robot will not be subject to major adversarial forces from the outside environment that significantly impairs its perception, navigation, control or motion.

Robot It is also assumed that the team shall be equipped with a μOrc board for control, two motorized wheels with encoders for movement, a camera for visual perception, a storage bin for transportation, and a robotic bulldozer arm with an incline for scooping. Moreover, these shall be in functioning condition. Access to a fully charged netbook from which the code can be deployed is also assumed. The robot will be powered by a 12V battery.

3 Approach

Low-level design There are three fundamental parts of the robot system: input from sensors, processing, output to actuators.

The robot will be equipped with a camera to provide visual information about the environment, and the encoders will record their ticks in order to estimate the robot's own position. Four sonars, one on each corner of the robot (left front, right front, left back, right back) will aid the robot in estimating its pose. Two bump sensors will be added to the front of the robot in order to ensure that it doesn't crash. The robot will also have a Break Beam sensor to determine when it's carrying capacity has been reached.

The code will both evaluate new sensor information and record state so as to guide the robot's output. The software is most suited to be built on the Robot Operating System (ROS) architecture, because of its asynchronous

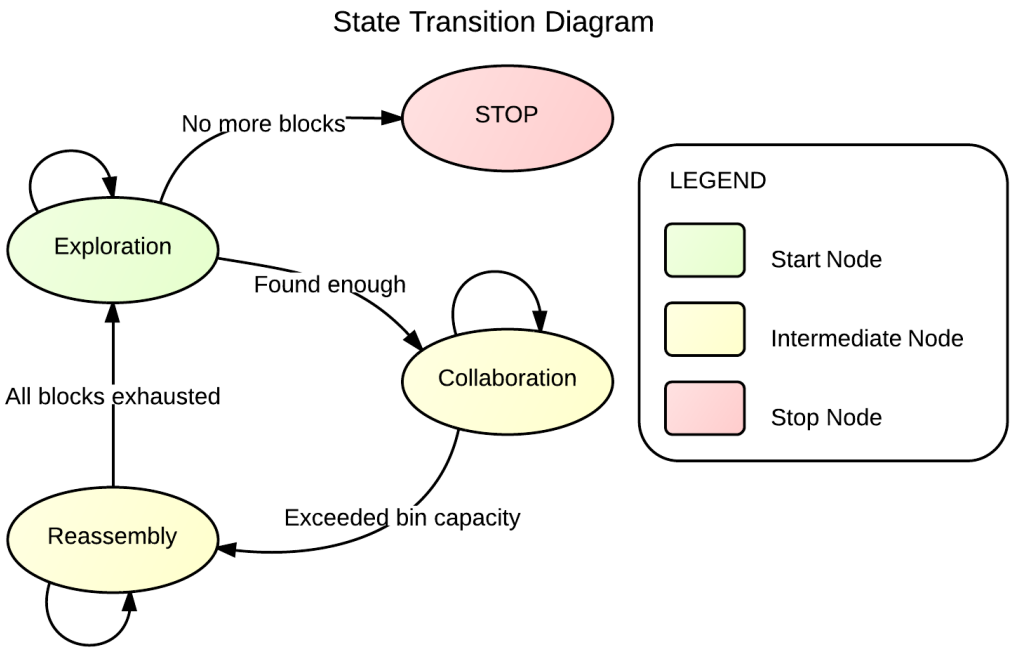

Figure 1: State Transition Diagram: State transitions in the robot's operation, robot can stop if it finds an emergency in any state

callbacks, complexity management through nodes and hardware agnostic behavior.

The output of the system will be actuated through the motors, which will be directly commanded by the processing's output.

High-level states The high level states of the robot as seen in Figure 1 would be: exploration, collection and reassembly. In the exploration state, the robot traverses the area using the mental map it has in mind with the aim of locating the building blocks. The optimal strategy is for the robot to explore the incomplete map it already has, visiting each place it thinks a block should be at, and updating its mental model should a discrepancy be found. In this stage, the robot should also keep track of the large empty spaces it finds, and note them as suitable for building the shelter.

The next state is the collection phase, where the robot travels up to or

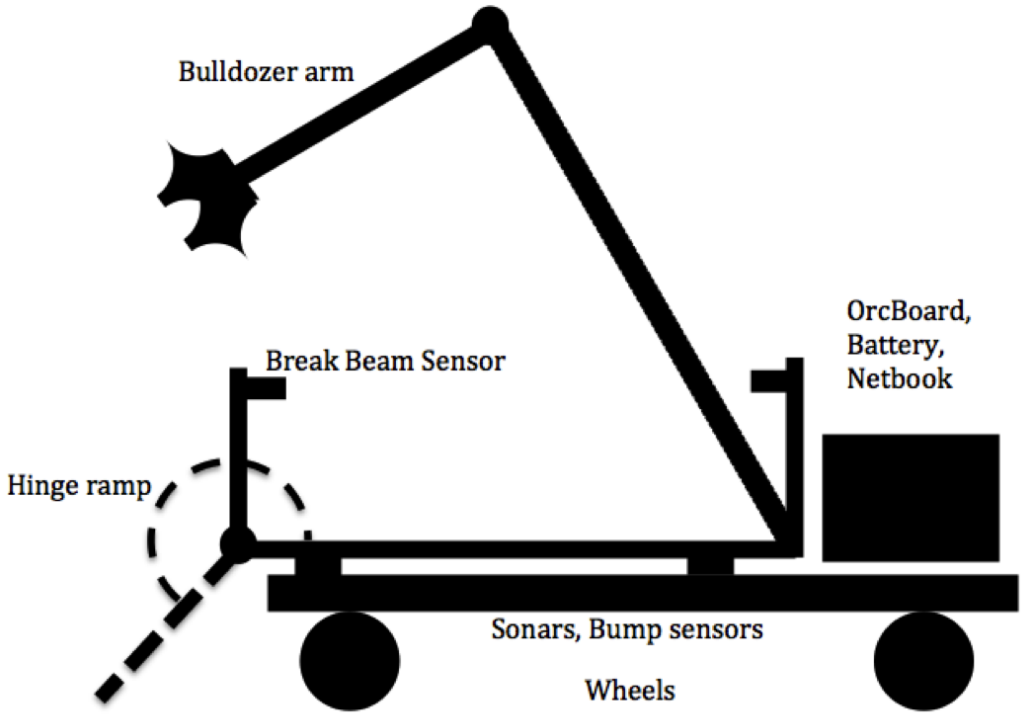

Figure 2: Robot Outline: Physical structure of the robot

close to the block, scoops it into its bulldozer-like arm into its bin. It will thus collect blocks until its capacity is full as indicated by the Break Beam sensor, or it has exhausted the number of available blocks according to its mental map (within a certain error margin).

In the final stage, the robot will drop the blocks flat onto the pre-planned position obtained from the exploration stage, using a hinged incline plane as the positioning device. Once the robot has exhausted all the blocks in its bin, it should go back to the exploration phase.

Note that if the blocks are of different sizes and the robot can detect that during the exploration stage, it should aim to pick up and drop the biggest blocks first so as to make the structure as stable as possible. Also, the robot has a time limit of five minutes per stage, to ensure that it does not get stuck in a particular stage due to unforeseen circumstances.

4 System Block Diagram

The modularity of the system can be represented by using a system block diagram, in which each subsystem is a node of the ROS. As can be seen in Figure 3, the system receives inputs from the Encoder and Vision nodes. The encoders will record ticks and feed the information to the Internal Map module. The Vision node, will employ the use of an OpenCV-like framework for object recognition, and feed the information it thus receives to the Internal Map. Vision is very error-prone, and one may need to carefully record an aggregate classifier of boundary lines, colors, shape recognition, distance in oder to reliably determine whether an object is present or not. This is also a good place to add memory to the system and record a few previous states, because it is likely that if the robot thinks it saw something in the last time step, the object will still be present in its field of view. The individual weights of these feature vectors in the final classifier can be parametrized by experimentation. The Internal Map node will also help guide the Object Recognition node in providing information about the *a priori* probability of a block being found at that location, based on the current map.

Internal Map provides location and terrain information to the State node, which additionally uses information about the robot's capacity to determine which state to put the robot in. The state transition diagrams are given in Figure 1.

State thus helps drive the Planning module, which will also use the Internal Map information to provide a path for the robot based on the robot's current estimated pose (location on x-y coordinate and bearing) and the relative positioning of the blocks. The Planning module will deploy a heuristic-based search, such as A star, to quickly determine the robot's route.

The Planning module gives instructions to the Grid Movement and Arm Movement modules, which actuates the motors. The Grid Movement could involve change of bearing i.e. rotation, or change of translational velocity. The Arm Movement in the Collection state will involve performing the entire series of steps that involve placing the bulldozer clamp over the object, rotating the side of the bin to allow the block to slide up the ramp into the bin. In the Reassembly state, the Arm Movement will involve rotating one side of the bin to become an incline plane again, releasing the shutter gate

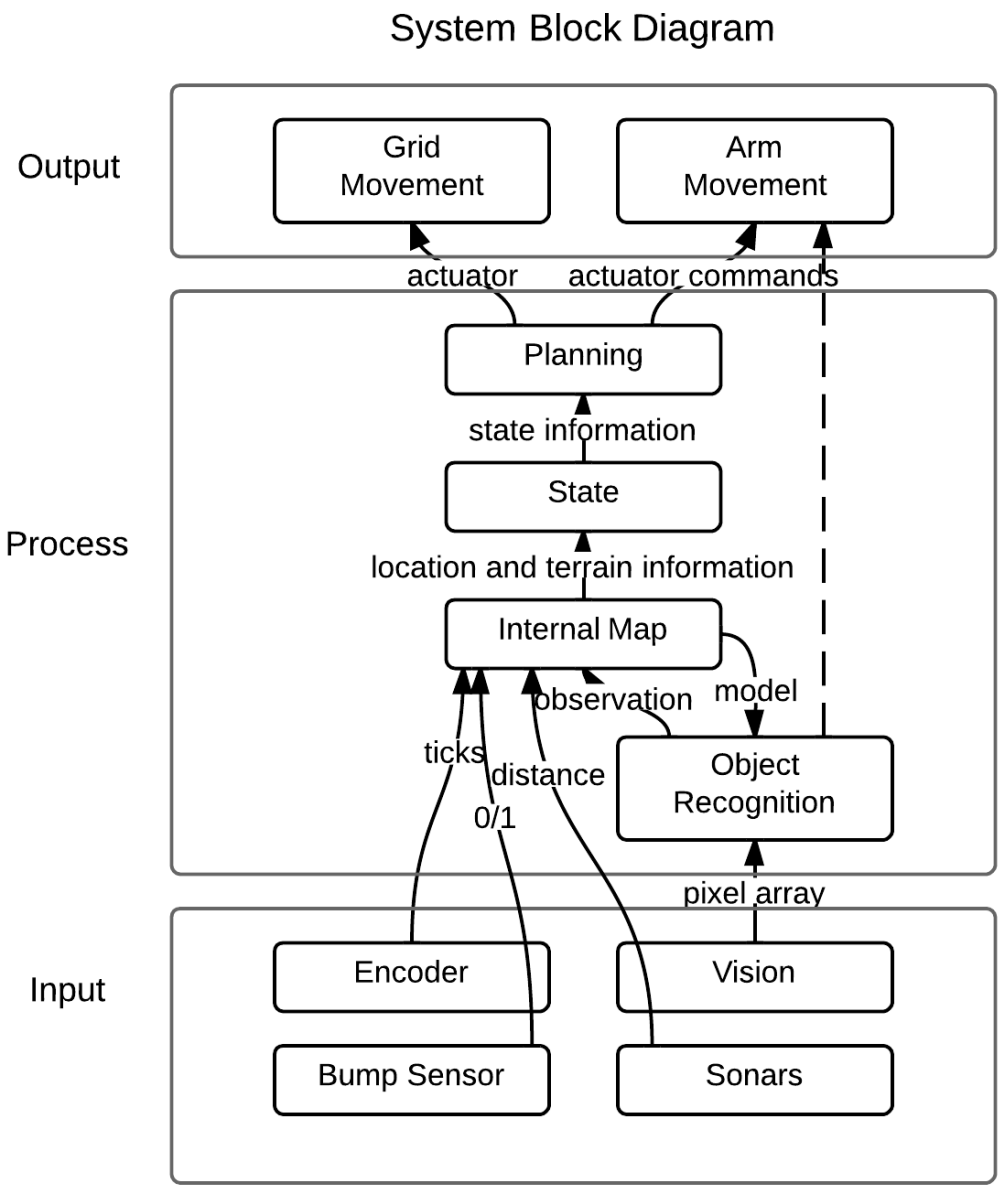

Figure 3: System Block Diagram: Communication between different nodes during the robot's operation

to allow the block to slide off the ramp into place. A bulldozer clamp and incline were used instead of a regular arm precisely because of the difficulty of reliable vision, which is needed for the precision of the robot arm. The motors will use a Proportional Integration Derivative (PID) control, which will allow for precision.

5 Errors

There is an emergency channel built into the system, where the direct input from Object Recognition will allow the motors to bypass all planning and stop should the module detect a collision. Information from the bump sensors and sonars will aid collision detection, in which case the robot will be commanded to stop immediately. This is expected to be used rarely, if at all, and the feedback system should provide for ordinary navigation control such as following a wall.

The wheels are prone to slipping and the sensors might not provide accurate information about the world, hence it would be necessary to build in error margins for most of the code. The robot might not calculate its absolute grid position correctly, but it could be resolved by adding the estimated errors (determined via prior experimentation) and determine an approximate error region where the robot could be, while taking input from the sonars and bump sensors to further narrow down the robot's real location. Thus, it is hoped that local observations will refine the global mental map of the robot through its course.

For the moving of blocks, the bulldozer arm accounts for minor errors by scooping up everything in its range. The reassembly of the blocks can be controlled because the blocks are being released from the robot's bin. In the case the map information is particularly damaged, the robot should rotate until it finds the nearest obstacle, find out what it is through vision (a block will appear as a differently colored blob, a wall will be plain), and if it is a block, just scoop it, whereas if it is a wall, it should align itself with the wall and follow it, while updating its mental map.

Table 1: Implementation Plan

Completion Date	Goal
03/12	Finalize initial design of modules and team contract
03/22	Build hardware of robot and calibrate
04/01	Implement Object Recognition and Internal Map
04/04	Implement Planning and Movement
04/07	Integrate using State
04/08	Improve Vision code
04/15	Integrate Vision with Movement
04/29	Iron out coordination issues
05/08	Grand Challenge

6 Implementation Plan

The implementation of the robot will be broken down into several stages and subgoals, which will serve as milestones for the project, and leave enough time for debugging and integration at the end. This can be seen in Table 1.

By Mid-March, we hope to have the design elements and team contract finalized.

In the next stage until end of March, the robot hardware will be built.

The first modules will be implemented through first half of April by integrating experience from the labs in terms of object recognition, localization, planning, state. Basic familiarity with ROS ensures that the robot can perform the right mixture rudimentary functions that were already present in lab. This combination might introduce novel challenges, and potential conflicts in data from different sources, which need to be resolved in the code.

In the second half of April, the integration of the sensors, processing and actuators will be emphasized and the robot will be tested extensively on different types of terrain and block conditions to ensure reliability.

In early May, the final aesthetic and fine-tuning touches are added to the robot, and we participate in the challenge!

7 Conclusion

It is anticipated that the greatest challenge in this project will be making the different subsystems and nodes reliably work reliably with each other, and enough time for the same in the Implementation Plan. Since the Robot Operating System lends itself to modularity, it will be easy to check whether an individual part functions correctly, allowing for faster debugging and in extreme cases, replacement of the entire module itself.

The Vision and Object Recognition node are the most prone to errors, and it is expected that this machine learning classifier-based approach will make the data more reliable by aggregating from different sources.

The challenge of perception, navigation and manipulation will require careful implementation of the modules on the software side, and correct wiring on the hardware side, with an intelligent interface that allows one to respond quickly to the other. The performance of this robot will be measured by the number of blocks it is successfully able to locate, collect and reassemble into its primitive wall shelter.