Jason Miller
2/18/2005
6.834J Problem Set #1

Part A

Fault detection and diagnosis: When operating in real-world environments, autonomous systems must be able to monitor themselves and react appropriately when a failure occurs. In some cases, the system may be able to work around the fault and continue. In others, it is sufficient to avoid making the problem worse and await further instructions. In either case, the system must be able to use limited numbers of sensors (which may themselves be noisy or faulty) to determine, first, that something is wrong and, second, identify the specific fault that has occurred. Both of these things can be difficult, especially given limited computational resources and real-time constraints. I would like to learn more about current techniques used to solve these problems in systems such as rovers and space probes.

Detection and tracking of moving objects: In the version of SLAM presented in class, an assumption was made that the world was static, i.e. that any changes perceived by the sensors were a result of the robot's actions. I would like to learn more about techniques that could remove this restriction and allow a robot to deal with other moving objects in the environment. These objects might be obstacles (like other cars on the road for a driving robot) or things to interact with (like other robots or the ball for soccer-playing robots). The first problem is to separate moving objects from the static background, especially when the robot (and its sensors) may itself be moving. The second problem is to track these objects and estimate their trajectories for purposes of avoiding, following or intercepting them.

Face detection and recognition in computer vision: To interact naturally with humans, it is helpful for robots to be able to detect and recognize faces. The first step is simply for the robot to recognize that it is looking at *some* face. This information can used to count or track people in its view since each person has exactly one face. In addition, a system such as an intelligent kiosk might want to know when a person is in front of it and looking in its direction. The movement of facial features can even be used to control aspects of a user interface. Even better than detecting that there is some person in a scene is the ability to recognize who that person is. This is useful for surveillance or access control but can also allow a robot to customize its behavior to an individual. Both of these problems are complicated by the fact that, in the real world, people tend to move around and may not be looking directly into the camera. I would like to learn more about current techniques to identify and recognize faces, even when they are misaligned or partially obscured. Some of these techniques operate on still images while others use video to aggregate data from a variety of angles.

Part D

R. Washington, "On-Board Real-Time State and Fault Identification for Rovers," In Proceedings of the 2000 IEEE International Conference on Robotics and Automation.

This paper was selected because it describes a fault detection and identification system for a rover-type robot. My robot would have appendages similar to what might be found on a rover and could therefore suffer similar types of failures. This system is based on the more traditional (than the next paper) techniques of Markov models and Kalman filters but attempts to fuse the two to reduce the computational burden of state estimation. In my robot, this work could be used to detect faults in the appendages or drive system.

The system presented is a hybrid of a discrete-state Markov model and a continuous state Kalman filter. In this context, the state space includes both normal states of the robot and states where one or more faults have occurred. Thus, by estimating which state the robot is in, we determine both that a fault has occurred and identify which fault it is likely to be. A Markov model is used to represent different global modes of rover operation (e.g. idle, driving, drilling). Different modes imply different "normal" versus "fault" behavior. For example, sensor values indicating a high drive current may be normal in some modes but indicate a fault in others. The Markov model state is then used to set the parameters and state space for a Kalman filter. The Kalman filter uses sensor data to estimate the continuous state within the Markov mode. In this way the system prunes the state space that must be estimated by the Kalman filter. The estimated state from the Kalman filter is then propagated back to the Markov model in the form of a simulated "observation" matrix.

However, when the system changes from one state in the Markov model to the next, it must carry over the continuous state to form the new initial conditions for the Kalman filter. Therefore, different instances of the discrete states are actually made unique by the continuous state fed into them. This effectively increases the number of possible Markov model states and can lead to an explosion of model size. Therefore, the system limits the number of Markov states to ensure that the computational complexity stays bounded. This means that some improbable states will be lost and it is possible for the state update formula to return a null state. They claim that this problem has been addressed in previous work but do not explain it here.

Based on their results, this system does a decent job detecting faults on a rover type robot and runs in a reasonable amount of time with limited computational resources. However, the system requires a large number of Kalman filter parameters and seems to be very sensitive to their values. It would most likely require a large amount of data to be collected from various trials of the robot to properly establish these values. In addition, the limitation on the size of the Markov model limits the accuracy of the state estimations. While this is clearly a limitation, it also allows for a trade off between computation and accuracy. If more computational resources are available, the Markov model can be allowed to grow larger, increasing it's accuracy.

V. Verma, G. Gordon, R. Simmons and S. Thrun, "Particle Filters for Rover Fault Diagnosis," IEEE Robotics and Automation Magazine special issue on Human Centered Robotics and Dependability, June 2004.

This paper was selected because it presents an alternative solution to the fault-monitoring and identification problem on rover-type robots. It is based on a substantially different idea than the previous paper thereby broadening my view of the topic. Again, the algorithms presented would be applicable to monitoring faults in the appendages or drive system.

Most traditional techniques for fault detection (e.g. the first paper) attempt to approximate the *problem* that needs to be solved in order to reduce computational complexity. In the previous paper, this took the form of limiting the size of the state space. This paper uses a technique called *particle filtering* to find an approximate solution to the complete, complex problem. The basic idea behind particle filtering is to use a Monte Carlo technique to pull samples ("particles") from the true solution. The distribution of the samples is then taken as an approximation of the true distribution. The main problem with particle filters is that a large numbers of particles must be drawn in order to accurately estimate the state when there are low probability states (e.g. fault states). This tends to negate the advantage of using particle filters in the first place.

This paper presents three techniques which allow particle filters to accurately model low probability states using fewer particles. The first technique assigns a risk factor to different states and uses it to influence the choice of particles. Fault states, which are usually low-probability but high risk to the mission, get a higher risk factor and are therefore more likely to be sampled. This helps to ensure that potentially serious faults will not be over-looked, even when using relatively small numbers of particles. The second technique groups together multiple fault states that have similar symptoms (like wheel failures) and treats them as a single state which now has a higher probability. If this merged state begins to become more probable (i.e. it appears that one of the faults in the grouping has occurred) it can be decomposed into individual states to find the exact fault. The third technique incorporates sensor data into the choice of particles, thereby making it more likely that states which explain those data will be sampled.

Using these techniques the authors demonstrate impressive accuracy using relatively small numbers of particles. However, they do not provide data on the actual amount of computation required to implement their system so it is difficult to compare it directly to the results from the previous paper.

J.L. Bresina and R. Washington, "Robustness via Run-time Adaptation of Contingent Plans," Proceedings of the AAAI-2001 Spring Syposium: Robust Autonomy. Stanford, CA.

This paper was selected to help answer the question of what happens after a fault is detected. The authors describe a command language and execution environment which allows for plans to be dynamically altered in response to failures or new opportunities without requiring a full-blown re-planning. It is especially useful in environments like the current Mars rovers where the robot is not capable of doing its own planning to begin with.

In this system, it is assumed that the robot receives a plan from an external source and then operates autonomously until its next scheduled plan upload. With existing technology, if the robot experiences a failure and cannot complete its plan, it will sit idle until it can receive instructions. Using the system proposed here, the uploaded plan would include alternate plans that could be executed if the robot detects that conditions were not as expected. This could allow a rover to try alternate techniques to achieve its goals or even switch to different goals if it encounters a difficulty with the original plan.

Two core techniques are presented, plan step skipping and alternate plan merging. Using plan step skipping, the robot would be able to skip over portions of the plan which are no longer reasonable and continue on with the rest of the plan, thereby achieving at least some of its original goals. For example, if a plan included taking several types of measurements but of the instruments had a failure, the other measurements could still be performed. Using plan merging, the robot would able to splice in a plan fragment either by inserting it between plan steps or replacing existing ones. This could allow the robot to retry an action which has failed, perform alternate actions if it continues to fail or perform extra actions if resources permit. The key to making this work it to assign utilities to different portions of the plan and the alternate plan fragments. Each plan step also includes conditions which must be met for the step to be executed. If the robot finds that a step cannot be executed, it can consider skipping steps or merging alternate plans to find the new plan with the highest expected utility.

This paper is an interesting extension to the status quo for rover control. It allows rovers to execute complex alternate plans by simply recalculating utility functions. While the resulting plans are more limited than you would get from a complete replanning, the reduced computational complexity makes it a good solution where computational resources on the robot are extremely limited.

Part E

The robot proposed in part B would be a very ambitious project. I believe that a manageable alternative would be to design one or two appendages, model them in a simulation and implement a fault detection system for those appendages. Since I would not have an actual robot, I would have to simulate failures and sensor noise. I could create a library of motion plans for the appendages by hand rather than implementing a planner.

I'd say the chances of my pursuing this project are about 40% but I would still like to pursue a project having something to do with fault detection and replanning.