Final Project: Route Planning

ENGS 31
Spring 2014
Thayer School at Dartmouth College

Demos by Appointment: May 27-28, 2014
Report Due: 9AM EST June 2, 2014

1 Background

Emergency rescue personnel are very concerned about optimum path planning to the site of a calamity.. When planning will want to select the most efficient way to move his soldiers from point A to point B. A rudimentary way this is done is via an estimation of the difficulty that troops will have moving over the terrain along their path. This estimation takes into account such things as steepness of grade, altitude of the location, soil type (sand, gravel, packed dirt, concrete, etc), risk of phyical harm, and exposure. We will refer to the effort required to navigate the terrain elements along a path as the *terrain navigation cost* (**TNC**).

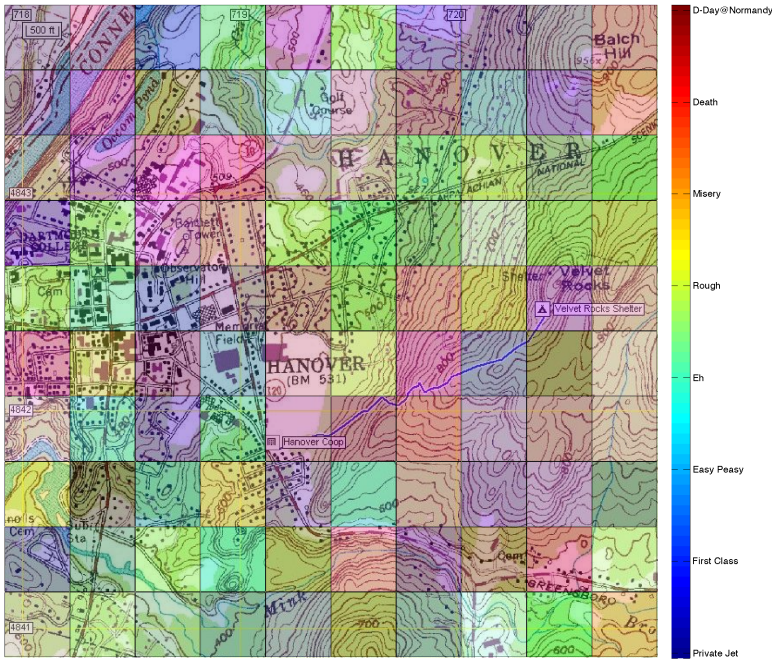

Figure 1: Example of a simplified, grid model of terrain navigation costs of the town of Hanover, NH. Values depicted are not indicative of the actual difficulty of traversing a particular terrain element.

Different systems of rating the TNC of a path exist but one of the simplest involves taking a top-down

projection map of the terrain, placing a uniformly spaced grid on it, and rating each resulting subsection with a numeric value representing the difficulty to navigate this region of the map. This addition to the map of a terrain region is shown in Figure 1. We can further simplify the problem by making sweeping generalization that they cost of traversing one of these regions is independent of the direction of travel or path chosen within the region. This is justified by our abiity to make the regions infinitesimally small such that the cost of traversing that region is uniform.

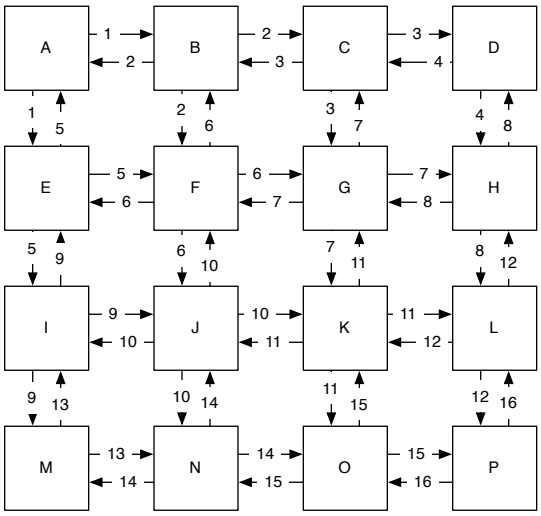

Figure 2: Four-by-four TNC grid represented as a directed mathematical graph. The costs of traversing each node from A to P increase from 1 to 16, respectively. These weights are shown as edge weights in the graph.

This grid structure that we have imposed forms regions with at minimum two(2) and at maximum four(4) neighboring regions in the cardinal directions: North, South, East, and West. For simplicity we define these as the adjacent neighbors which we can reach from a particular region. This format allows us to consider this representation of the map and TNC as a weighted mathematical graph with each region represented as a node in the graph and the cost of traversing from a region to an adjacent region as weights on directed edges originating at the starting region and connecting to the destination region.. An example of this casting of the problem can be seen in Figure 2.

With these simplifications we are able to begin strucutring an algorithm to determine the minimum sum TNC for travelling from a source to a destination across a map with a square grid of fixed size. Broadly speaking, there are two classes of algorithm to solve this problem: single-process and distributed. You may already be familiar with some of the single-process solutions to this problem, they form a class of algorithms known as tree search algorithms. A distributed solution to this problem is less well-studied, however Professor Fossum holds US Patent 5548773 A which describes a way to use independent computation units to find an optimum path.

2 Problem Statement

Leveraging the structure we have developed in Section 1 you will design and implement, as an individual project, a digital logic system on the Digilent Nexys 3 board which will find an optimum path from a start location to an end location on a $16x16$ grid. The values of the TNC for each region, the region specified as the start location, and the region specified as the destination will be communicated to the FPGA using RS-232 serial communication from a computer. The cost of traversing each of these cells will be shown on a VGA display. The FPGA will wait for a start signal from the serial port and then begin search for the optimum path from the start to the destination, updating the VGA display as it progresses. The FPGA will continue program execution until either it has found an optimum route or received a stop command

from the serial port. Once stopped the optimum or completed route will be calculated and displayed to the user on the VGA display. Finally you will document your project in a handsome report that will be handed into the course staff. This report should be comprehensive and include your approach to the problem, your design documents (FSMs, RTLs, block diagrams, etc), source code, module simulations, and evidence that it actually worked when instantiated on the FPGA.

As a reach goal we would like you to transmit the optimum path back to the computer via the RS-232 serial port. The format of this communication will be defined at a later date as we gauge student's progress with the project.

This is meant as both a design and implementation exercise so while the course staff are happy to discuss design strategies, principles, and with you we are leaving it to you to structure the way you want to tackle the task. In Section 2.1 below we have described the interface with the computer for sending data from the computer, however beyond that we are deliberately leaving the task underspecified to allow you the maximum in design freedom. We are also providing Dave's implementation of the VGA grid which should speed along the process of designing the system, however if you find its structure clashes with your design feel free to modify or discard it.

The staff have made a simple python script that will be made available to test your system end-to-end but we **strongly** encourage you to design using the best practices emphasized in the course and unit test your modules through simulation as you go rather than as a monolithic unit and integration test. If you are having difficultly using the script to test your program feel free to email Harrison and he will walk you through its operation.

2.1 Serial Communications Protocol

Serial communications between the computer and FPGA run at 115200 baud with 1 start bit, 8 data bits, 0 parity bits, and 1 stop bit with no flow control bits. The bits are transmitted in LSB order.

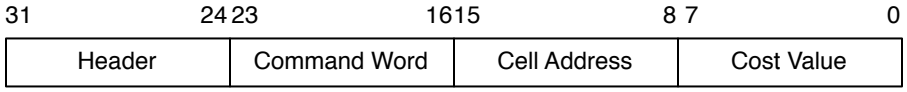

Figure 3: The structure of a data packet for loading the FPGA with a new TNC values.

Table 1: Control words for the FPGA-loading data packet shown in Figure 3.

Header		0x57
Command Word	Cell cost write	0x01
	Beginning Address	0x02
	End Address	0x03
	Start Program Execution	0x04
	Stop Program Execution	0x05
	Invalid	0x00 and 0x06-0xFF
Cell Address	*valid only with command words 0x01-0x03*	0x00-0xFF
Cost Value	*Valid only with command word 0x01*	0x00-0xFF

3 Parting Guidance

This sounds like a daunting task, and it is, however you have many of the components for this already built from Labs 4-6 and should be able to leverage them in your designs.

The way you approach the problem from the very beginning will drasticallty change your implementation of the project. Should you choose to implement a tree search algorithm such as A^*, Dijkstra, Depth-first

Search (DFS), or Breadth-first Search (BFS) you will spend your time worrying about memory limits of the device and may have to use one of the Nexys 3's on-board, but off-chip, memory resources however you may get your path for free. Should you choose the distributed computation approach performing the search will be more straightforward, however the method of recovering the path you traversed will be more difficult and you may find you are resource constrained for the signal fanout required to recover the optimum path.

Get started on this as soon as possible. The course staff are here to help you make this project happen, but we can't make magic happen if you haven't spent time thinking about an working to implement this project. We would love to meet with all of you sometime in the week of May 11-17 to make sure you are on track with your designs. As this is an individual design project we will have no idea how you are chosing to tackle this problem so please bring block diagrams, RTL diagrams, FSMs, HLSMs, or pseudocode so we can be as helpful to you as possible.

Remember you will be documenting your project with a report even if you are unsuccessful in completing it. Thus it behooves you to have liberal documentation of your design process. We recommend iterating through high-level diagrams quickly initially then begin documenting them digitally with a charting program or, at minimum, scanning your hand-drawn diagrams.

You are free to discuss approaches, your design, and help each other debug, but write your designs and VHDL on your own.