Upload of Dead Reckoning Measurements for Improved Navigational Efficiency on Embedded Robotics

Andrew J. Tickle and Paul K. Harvey

School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, Liverpool L69 3GJ, United Kingdom

Email: prouction_leader@hotmail.com

Abstract. The process behind Dead Reckoning (DR) is simple in that a robot can know its current location via a record of its starting position, direction and speed without the need to look for landmarks or follow lines. This process allows a robot to drive around a known environment such as indoors and heavy urban areas where traditional GPS navigation would not be an option. Discussed in this paper is an improvement of a previously designed DR mechanism in DSP Builder where now the user enters the DR measurements and commands as a sequence via a keypad. This replaces the need for user to programme the details into the system by altering numerous value tags within the design one-by-one, thus making it more user-independent and easier to alter for different environments. The paper shows updated simulations for repeatability, how the keypad links to the system and where this work will lead.

1. Introduction

The material presented in this paper continues the work presented in [1] and uses all of the same assumptions (such as no wheel slip), the same simulated test environment (first figure in [1]) and the same robot parameters (such as the wheel diameter and step size of the motors). The way that this is achieved is by modifying the previous design so that each of the counters and comparators can be easily adapted so that their inputs can be obtained from an external source. In this case a keypad will be used to simulate the inputs to the system in order to enter the Dead Reckoning (DR) measurements and commands into the system. The design will follow the same general design methodologies and simulation procedures as presented in [2].

The main reason why DR is used is as it is a well proven method for robotic navigation when GPS or landmark based navigation is not effective, such as in indoor environments or when the robot is required to be small and compact. The mathematics behind DR differ from robot to robot due to physical parameters and the type of steering mechanism used. The two main steering types used in conventional mobile robotics are either a car-like steering system, this allows the robot to turn in a curvature of constant and known radius, whilst a differential drive robot (the case considered in this paper) can make a very narrow turn and spin on the spot. The process of knowing its current location deals with two variables: The distance travelled by the robot ($x$) and the angle ($\theta$) through the directional axis of the robot, this means that the robot will travel a set distance $x_1$, then orientate itself by $\theta_1$ before moving along its next distance, $x_2$ [3]. This process is repeated many times until the robot

---

1 Andrew Tickle is now a Lecturer in Electrical Engineering at Coventry University
reaches its final position where it will spin on the spot and then return to the starting position by retracing its steps.

2. Modifications to the DSP Builder Dead Reckoning System

Several of the blocks from the original system will remain unchanged by these updates as they do not require any further modification. These are the independent calculator that allows the speed of the motors to be calculated by using the onboard clk from the FPGA and the Pulse Width Modulation (PWM) blocks that create the signals to be sent to the stepper motors. Therefore, the only existing block that needs modification will be the DR navigational protocols and a new keypad interface block will be added to allow the keypad data to be processed and sent to the DR block. The new Register Transfer Level (RTL) [1] diagram is shown in Fig. 1, however the data validation has been omitted on this figure as it outside the context of this work. The new keypad interface block connects the multiple signals that will send data to the counters that make up the DR navigational protocols in the form of a bus, the inputs to this block are the keypad’s numerical data, stating the distance to travelled every odd value followed by a command operation for every even operation, the total number of commands and distances entered and an “OK” signal to allow the DSP Builder system know when the value has been entered. The operations are as follows; the old Movement E operation to stop the robot has now been hardwired into the code and so is not included in the list of usable operations:

- **Command 1 (Movement A):** The robot performs a point turn left.
- **Command 2 (Movement B):** The robot performs a point turn right.
- **Command 3 (Movement C):** The robot performs a spin on the spot.
- **Command 4 (Movement D):** The robot will increase its forward speed and deactivate object avoidance sensors.
- **Command 5 (Movement F):** The robot will wait a preset period of time.

Therefore when Command 3 is entered into the system, the robots DR sequence is complete as it will spin on the spot and travel back along the route it has just travelled. When this command is entered by the keypad it will send in the total number commands and distances into the system, this value is used to determine the return route for the turning commands as if the robot turned left at a corner before the end of the sequence, then on the return route it would have to turn right at that corner. As a result, a useful equation has been developed to perform this function:

\[
\text{Return Operation} = (\text{Total No of Operations} - \text{Operation No}) + \text{Total No of Operations} \quad (1)
\]
The above equation allows the location of the inverse movement to be calculated and correctly inserted into the system. Now looking at the keypad interface block shown in Fig. 2, this is a screenshot from the DSP Builder environment but shows only a fraction of the circuitry as the whole system is too complex to understand from a single screenshot. The data from the keypad enters the system where the distance parameters shown in the top of the figure, have value tags of ‘1’, ‘3’, ‘5’ etc linked to comparator blocks in order to detect these odd values from a counter which uses the “OK” signal to keep track of the entered values. If an odd match is found then this is multiplied by a coefficient of 90.9 obtained from [1] that will convert the distance into a step pulse for these given robot parameters. This value is then held via an infinite OR-loop (an OR gate with its output feeding an input) before being output to its corresponding sequence block, this is directly fed into the comparators on the output of the counter to replace the value tag that was manually entered on the previous version. The distance inputs have a one-to-one relation with counters in the DR block as the counters will need new distance measurements and so cannot be easily reused. The operation commands on the other hand can easily be reused, this will reduce the amount of counters in the system and in the end, save resources on the FPGA for when the system is ported into VHDL for full system implementation. There are five commands blocks, each one representing an operation from the bullet points shown previously. Moving back to Fig. 2, these commands (two of which are shown) can be seen in the bottom part of the figure, these look to see if the counter produces an even result and if so, the product lets the data pass through to the next set of comparators where it searches for a match to the entered value and when one is found that output line goes high. The counter from the keypad “OK” signal is also output and sent to the DR block.

Figure 2: Gate level diagram of aspects of the keypad interface

The process by which the command operations are assigned is slightly different to that of the distances as all are used by the same operation block over and over again and values are stored in infinite OR-loops that identify the position in the sequence to be used by the master clock to activate that block when it is required in the sequence of movements. Each of these operations has the number of steps pre-coded into it via a value tag as in [1] and just uses a multi-input OR gate to send the information into the actual count mechanism of the operation. The overall process is displayed in Fig. 3(a) to avoid the confusion of a graphical approach, this can be done numerous times for different sequence positions via a multi-input OR gate and a number of infinite OR-loops retaining sequence values that feed the multi-input gate. The master clock remains largely the same as it did previously apart from now, like all the other counter blocks, is set to a very high value unlike previously and has a comparator set to ≤ so that it does not exceed the total number of operations. The comparator limits
on the second counter have an upper limit of the total value so that after this value, the "updown" signal on the first counter will fall to '0' and it will start to count down and thus allow the robot to retrace its steps. This can be easily visualized by looking at figure 7(b) in [1]. The overall structure of the DR block can now be seen in Fig. 3(b), the blocks at the top represent the numerous number of blocks that can have distances sent to them for travelled forward, the blocks below that are the command operators, the blocks in front of them represent the selection process described in Fig. 3. The latter is also linked to some mathematics blocks that perform Eqn. 1 using values from Commands 1, 2 and the total number of operations so that the reverse sequence can be completed. Finally, like in the previous system, the master counters send and obtain data from all counters in the sequence to ensure the sequence of events is kept, the reverse turn operations have two signals from the master counters as one is from the second counter for when the values are over the total in order to override the original command and make it turn the opposite way on the return journey.

3. Results and Keypad Interface

The keypad was always going to the intended way of entering the data into the system, the difficulty was deciding on how this data would be used. The initial method investigated was to just input a long string of data and using a series of comparators or bit extractors in order to extract sets of numbers from this in order to make the numbers for the counters. This however failed to live up to expectations and so another method of using dividers to reduce the data down was devised, this was partially successful but the results still suffered from data contamination. Finally the chosen method was to enter a value into the keypad and press the "#" key before entering the next value was devised, this also allowed for easy interchange between distances and command operators.

The keypad code has been designed to interface directly with the DSP builder circuitry, and therefore can be used as a complete user interface solution for implementation onto a physical device. The top-level block diagram of the keypad can be seen in Fig. 4. The keypad completes the design by providing input error checking and entry cancellation via a dedicated cancel key (star "*" key). The design as shown in the aforementioned figure occupies 250 logic elements, of which takes only 3% of available resources on an APEX 20K200EFC484-2X FPGA, and may certainly be further cut down by removing the test and debugging outputs. This portion of the robot design works by first scanning for input keys and stores the keys into a synthesized RAM buffer. The circuit checks for overflow on the input and also clears the buffer on the keying of the 'clear' key. The keypad design can be seen
working in Fig. 5(a), where the system correctly registers the distance key presses and converts the binary-coded decimal key buffers into a single 10-bit word that can be directly passed on to the DSP-builder section of the design, giving the robot a range of 1023 m per movement command, with a resolution of 1 m. Note that for the purposes of the simulation, the keypad scanner was removed and data entered directly into the main state machine. When interacting with the real-life implementation, the keypad scanner was used.

Figure 4: Top-level block diagram of the keypad interface.

4. Conclusions and Future Work
As seen from the experimental results, the keypad and the robot simulations work effectively together in order to help programming the DR aspect of a mobile robot easier for the non-expert by means of an easier keypad interface. One aspect that will be changed in future versions of this work will be to use the distance measurements in millimeters rather than meters, this would increase the bit usage from 10-bits for the current range of 1023 m to 20-bits and then this will provide more accurate robot movement in the real world whilst still lying within the 50-bit operation range within the DSP Builder software. Then by a simple process of expanding the registers to handle larger numbers to therefore increase the operational range of the robot. Future work will consist of adding stereo vision based navigational techniques into the design for object avoidance as well as PID controls for compensation of errors such as wheel slip which have so far not been included in any designs. By including these, the mobile robot envisaged will be more robust and adaptable to changes within its environment and be a better tool for applications such as security, condition monitoring and human-robot interactions.

Acknowledgements
Dr Tickle and Mr Harvey would like to thank the University of Liverpool for providing the resources to undertake this work and to Prof. J S Smith and Em Prof. J Lucas for their support and guidance during the course of this work.

References
[1] Tickle A J, Meng Y and Smith J S 2010 Simulation of a Dead Reckoning Embedded System Security Patrol Robot for Deployment inside Structures and Buildings Proc. SPIE 7833: Unmanned/Unattended Sensors and Sensor Networks, Toulouse, France, September 2010 (12pp)
[2] Tickle A J, Smith J S and Wu Q H 2007 Development of Morphological Operators for Field Programmable Gate Arrays Proc. IOP Conference: Sensors and their Applications XIV, Liverpool John Moores University, Liverpool, UK
Figure 5: (a) Simulated keypad signals, showing the system reacting to the entry of two distance values and two commands, (b) MATLAB plotted results from DSP Builder for the robots route in the virtual environment.