

IMPLEMENTATION OF A SIMPLE MOVING TARGET INDICATOR USING LABVIEW

GARRETT WARNELL
ECE 480, FALL 2008
TEAM 4

ABSTRACT

This paper discusses the digital implementation of a simple moving target indicator (MTI) using LabVIEW. A single delay line canceller is chosen for study due to its simplicity in both concept and implementation. A modification is made to facilitate implementation in a common embedded systems environment, and an example LabVIEW program is developed for demonstration.

KEYWORDS: moving target indicator, MTI, filter, radar, digital signal processing, DSP, LabVIEW, delay line canceller

INTRODUCTION

For radar and other similar systems, it is often desirable to be able to distinguish between moving targets and those that are stationary. Such information can greatly aid operators when the surveillance targets of interest, such as vehicles or people, are located in environments with a high amount of clutter (undesirable return signals from the surroundings). Fortunately, through the use of digital signal processing, it is possible to extract this information from the system in real time using a moving target indicator (MTI) filter.

OBJECTIVE

This paper will focus on the development of a simple MTI filter using LabVIEW. A modification will be made to a classical filter in order to facilitate use in a non-ideal embedded system. The exact system and the method of signal acquisition will be considered arbitrary, though it is assumed that data will arrive in successive blocks, each representing some sort of pulse. Further, it is not the objective of this note to introduce the user to LabVIEW, and basic knowledge of the software will be assumed throughout.

BACKGROUND

One of the simplest methods of moving target information extraction is the use of a single delay line canceller as an MTI filter. A block diagram of the system is shown in Figure 1 below.

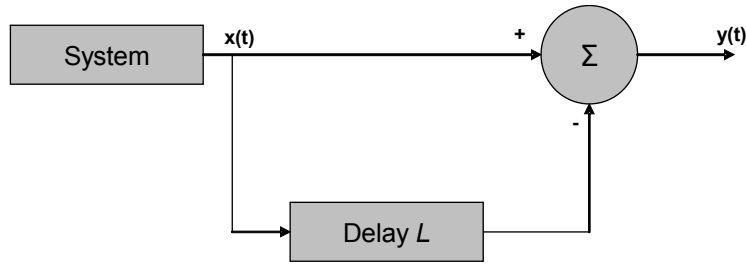


Figure 1

Assuming a pulsed radar system is used (i.e. the RF energy is transmitted periodically for discrete amounts of time), if the delay L corresponds to the time between pulses, then the output $y(t)$ appears as the error between consecutive pulses. Therefore, the output only contains information pertaining to changes in the surroundings that have occurred since the last pulse, i.e. moving targets.

The single delay line canceller shown above can also be described by the discrete time equation

$$y[n] = x[n] - x[n - L]$$

Equation 1

While other more advanced MTI filters exist, the filter described by Equation 1 will be the main focus of this paper.

SINGLE DELAY LINE CANCELLER MODIFICATION

The concept of a single delay line canceller MTI filter can be extended in order to facilitate its use in a more realistic embedded system. That is, this filter can be modified to work in a system in which pulse data arrives in discrete blocks with an undetermined amount of time between successive pulses. Since the purpose of the filter is to subtract the previous pulse observation from the current one, it will be sufficient to simply store pulse data in memory and use this data in the modified implementation of the MTI filter. A block diagram of the modified filter is shown below.

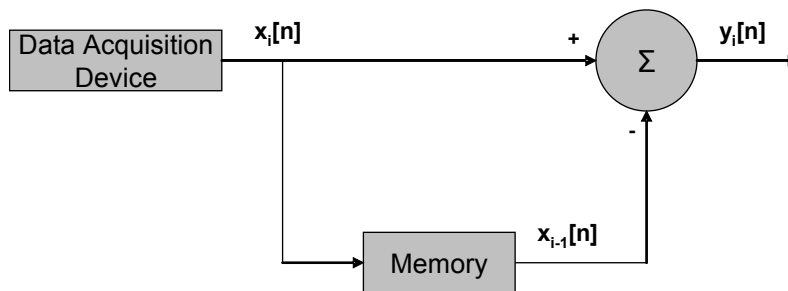


Figure 2

The signal $x_i[n]$ represents the finite duration digital signal containing observation data from the current pulse, while $x_{i-1}[n]$ is the same signal from the previous pulse.

Thus, the values of the continuous input $x[n]$ between pulses are no longer of importance. Likewise, knowledge of the delay L is no longer necessary. This information is in essence replaced by the assumptions that each pulse is of exactly the same in length and was captured in exactly same manner as its predecessor. These requirements are easily realizable in many systems, and such a system is assumed for the following discussion.

LABVIEW IMPLEMENTATION

The modified MTI filter discussed above can be implemented without much difficulty using LabVIEW. The structure shown below in Figure 3 is assumed as a basic starting point for a system that might make use of such a filter. A basic `while` loop facilitates the continual acquisition of sweep data, and sends the filter output to an arbitrary location of interest.

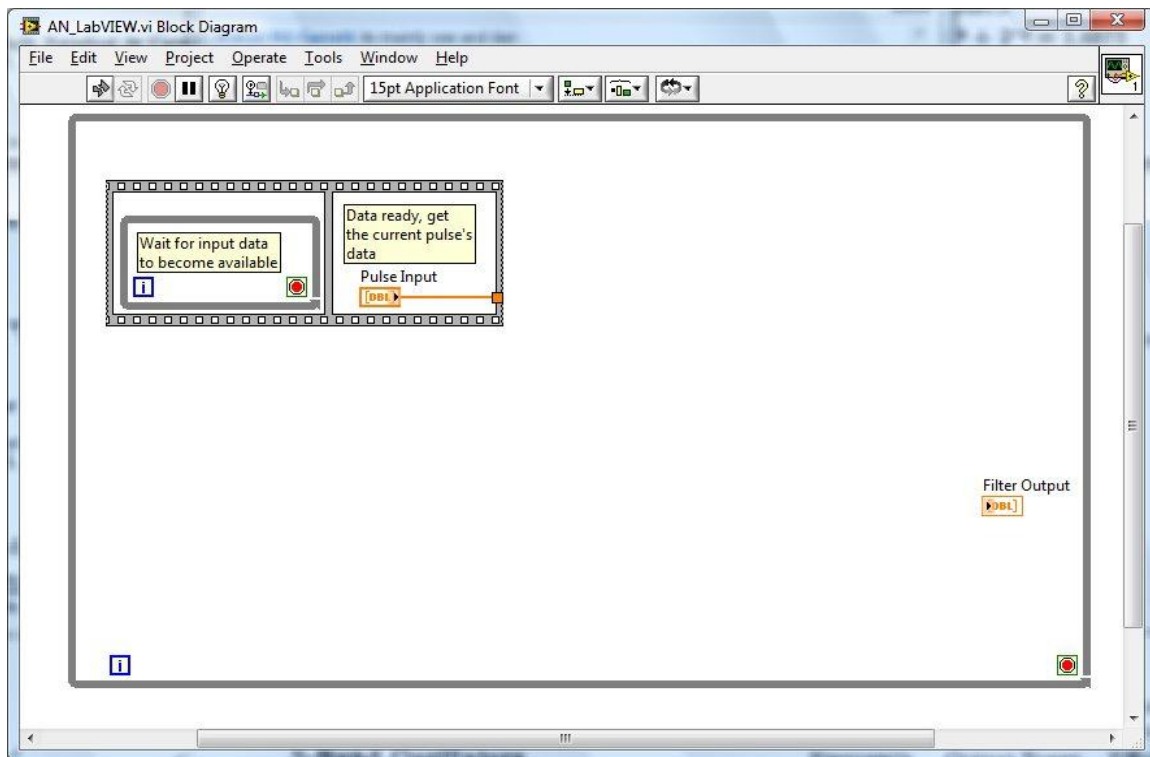


Figure 3

The next steps in implementing the filter are summarized below.

1. Allocate memory for system
2. Perform subtraction (filtering)
3. Perform iterative update of system memory

To allocate memory for the system, an initial value must be chosen. It is best to choose a value of all zeros for the first run, as subtracting this is equivalent to the system output without the filter. For successive pulses, the memory will be updated and the filter will perform as expected. For efficiency, memory should only be allocated during the first loop iteration, and then overwritten during successive iterations. The allocated memory must be of the same length as the expected pulse data. Here, it is assumed that each pulse will be represented by 1024 samples. The memory is also assigned to an indicator so that it may be referred to as a variable for the remainder of the program. Figure 4 shows one method by which to achieve this memory allocation.

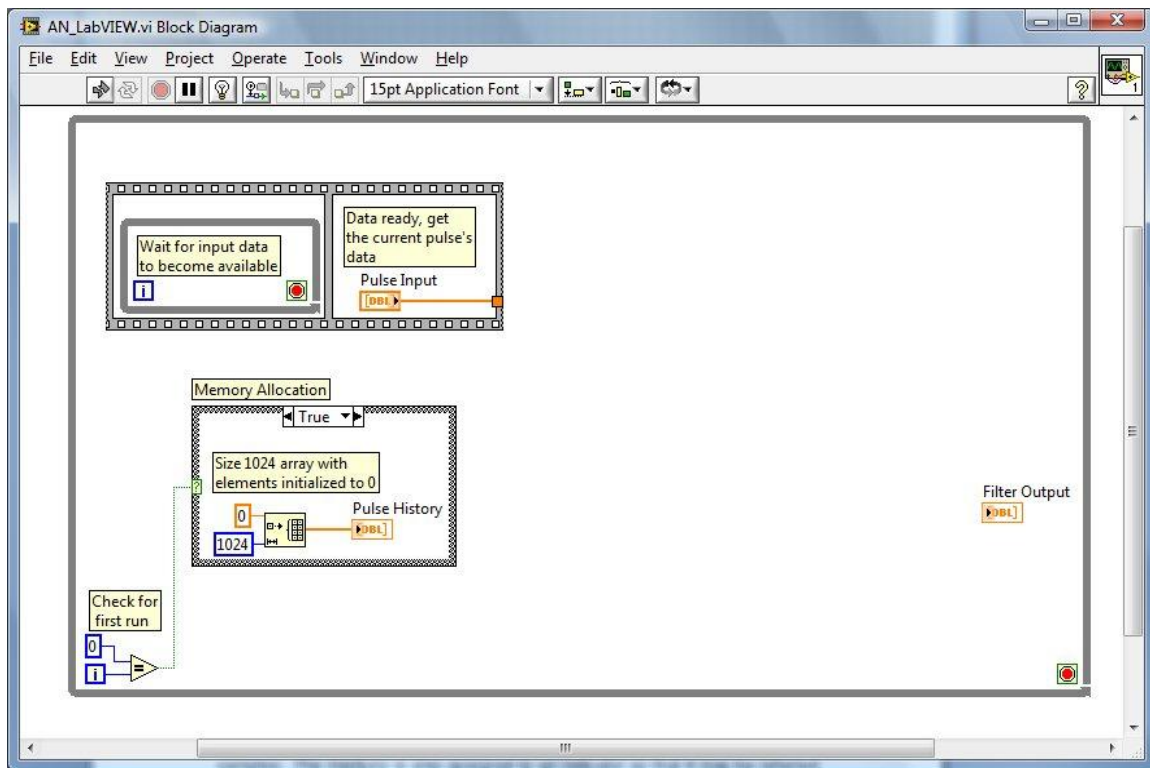


Figure 4

Now that the memory for the system has been allocated and initialized, there needs to be functionality by which to subtract the values in memory from the current pulse data. This is done by referencing the variable pertaining to the system memory and utilizing the LabVIEW subtraction block. The output is then passed to the filter output block. This implementation is shown in Figure 5.

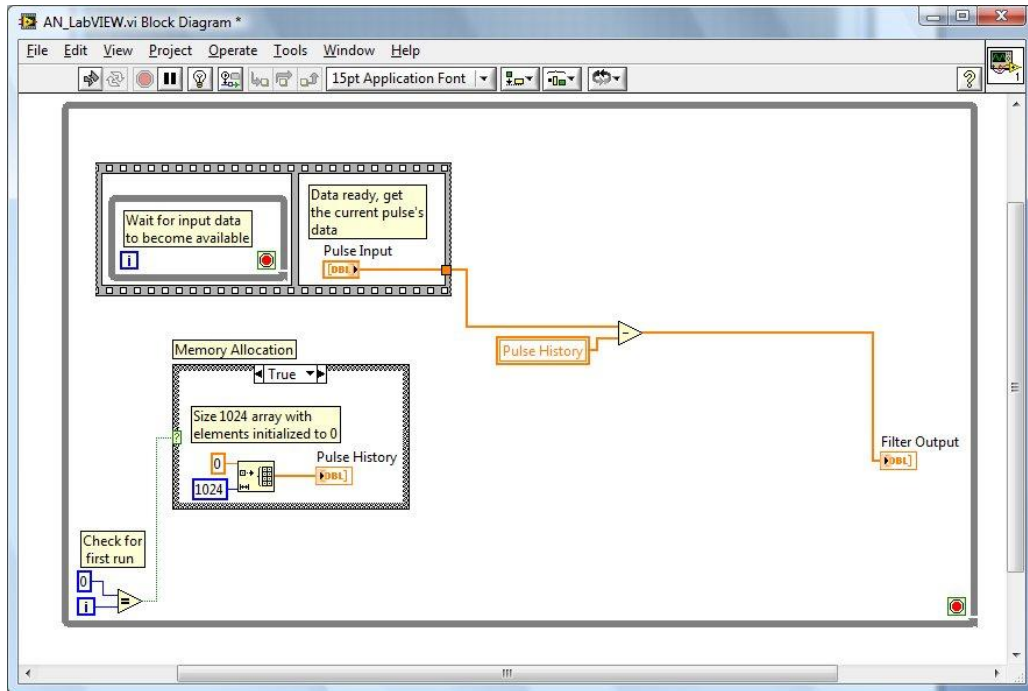


Figure 5

Finally, at the end of each iteration, the values in the system memory need to be updated to reflect the data from the current pulse. These values will be used during the next iteration as the “pulse history.” This update is done through a reference to the system memory. It is also necessary to place the subtraction and update functions in a sequence structure to guarantee the sequential order of “subtract, then update.” This is shown in Figure 6.

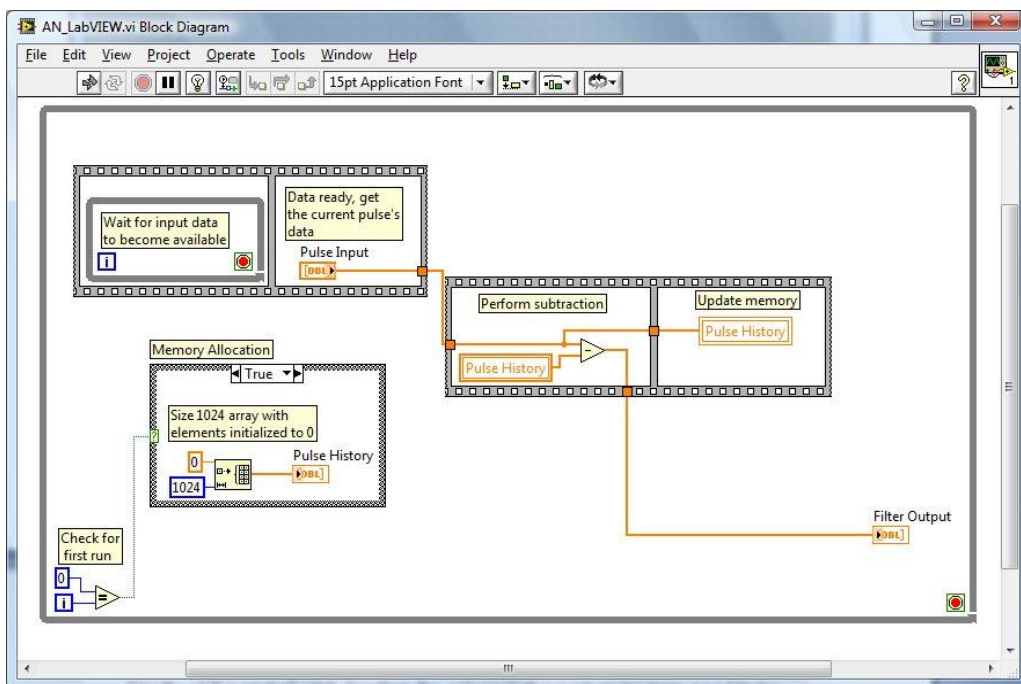


Figure 6

CONCLUSION

A simple MTI filter can be implemented as a slightly modified single delay line canceller in LabVIEW. This is done with a simple system that iteratively updates a memory block. The values in memory are subtracted from the current input, and a filtered output is produced. For further development, a threshold could be applied to the filtered output in order to search for signals that were able to survive the system. Signals that are passed indicate moving targets. If such a signal were found, an alert could be generated to indicate such to an operator. While the user may wish to employ more advanced MTI filters, the above gives an introductory overview of how LabVIEW may be used to realize such a filter.

REFERENCES

1. Levanon, N. and Mozeson, E. Radar Signals. Wiley-IEEE, 2004.
2. Zverev, A (1968). *Digital MTI Radar Filters*. IEEE Transactions on Audio and Electroacoustics. 16(3), pp. 422-432.
3. Mark, J (1972). *A Recursive Digital MTI Radar Filter*. Proceedings of the IEEE. June 1972.