A Graphical Framework for Constructing and Executing Computational Networks

Christopher L Hall
Consortium Research Fellows Program
Dayton, OH, USA

Vincent A. Schmidt
US Air Force Research Laboratory
Wright Patterson Air Force Base, OH, USA
Vincent.Schmidt@wpafb.af.mil
(Corresponding author)

Abstract— **Research in multispectral data visualization frequently consists of experimenting with combinations of a variety of fusion and visualization algorithms. This paper describes the design and development of a flexible GUI-based software utility that can be used to rapidly construct networks of configurable filters to be used in multispectral visualization research.**

Keywords-model execution; simulation

I. Introduction

Research in multispectral data visualization frequently consists of experimenting with combinations of a variety of fusion and visualization algorithms. Formal experiments generally include evaluation by human subjects, and the mandatory approval processes for conducting these experiments can be long and tedious. In addition, the system design and programming effort required to generate the specific software used in the experiments is also time-consuming, even though the primary difference between experiments is often only the image stimuli and the selection of fusion and visualization algorithms. It occurs to us that considerable time and effort could be saved if the experimenter had the ability to easily and quickly evaluate potential combinations of algorithms (we alternately refer to as "blocks" or "filters") without heavily relying on software personnel.

Thus, it is beneficial to be able to rapidly construct a network of algorithm building blocks, each with its own set of defining parameters, as a preliminary experiment prior to full-scale study development. This paper describes the design and development of a flexible GUI-based software utility that can be used to rapidly construct models (networks of configurable filters) to be used in multispectral visualization research. An experimenter with little or no software development experience can use the building blocks to graphically construct and test a specific filter network configuration, and software personnel could assist when the final configuration is selected, if required. This allows the experimenter the freedom to investigate optimal or interesting filter settings and combinations of filters prior to committing to developing specific filtering software to use in an experiment. (As an additional benefit, the GUI-based application can also be used by intelligence analysts to construct interesting and useful filter networks for examining individual frames of multispectral imagery.)

II. Concept Description

The concept behind the software allows the GUI tool to import a variety of models without loss of generality. As a proof of concept, the original models were developed as simple binary logic design operations (AND and OR gates) and were used to demonstrate logical operators on input files containing lists of binary vectors. Within a single day, a small set of filter functions and models were developed for basic manipulation of JPG image files, demonstrating the versatility of this design for use in multispectral image analysis.

The user graphically adds and connects blocks within a network graphically, where each block represents a specific algorithm (filter). The set of filters is fully dynamic and customizable. New filters can be easily added by generating appropriate software code into the filter files. The network connectivity between filters is intended to be very general, allowing filters to have multiple inputs, multiple outputs, and individual outputs connecting to multiple filters. (The current execution mechanism does not allow for recurrent, or cyclic, networks; only feedforward networks are currently supported.)

Filters are not limited to single inputs (such as binary values or individual JPG files), but can also be programmed to accept multiple data inputs. For example, a block that accepts three frames of multispectral imagery might implement a data fusion algorithm, or a block that takes two JPG image frames as inputs might output the frame with a higher information content. As another example, consider a filter that takes multiple images (from different viewpoints) as inputs and generates a 3D model.

Specific filters could also be designed to operate on data streams. For example, a filter might accept a stream of UAV imagery and perform video stabilization operations, outputting the stabilized stream. Another filter might simply read an image stream and output individual graphics frames, or input a stream of audio and output the stream with noise reduction applied.

The design also allows for each graphical building block to display intermediate results as data passes through the network. The display mechanism can be selected by the user (e.g. actual image, histogram, EXIF data, explicit value, etc.) from a list of display options. New display functions are easy to add, limited only by the complexity of implementation.

III. Implementation

The entire application (including filter blocks and display functions) is implemented as a collection of functions written in the cross-platform Python programming language. Python is a byte-code-compiled object-oriented scripting language with outstanding support and a robust set of development features and libraries. The software can be executed in Microsoft Windows, Linux, or any other operating system supporting Python and the Gtk graphics libraries. If needed, algorithms written in other computing languages can be directly integrated into the application using mechanism available through Python.

Executing the application consists of loading the model and indicating the input source(s). Inputs are cycled sequentially through the network, with intermediate results being displayed (when this feature is selected) as execution occurs. Final outputs can be written to files or simply displayed to the user.

Steps can be taken within the model to ensure validity and applicability of inputs to each filter as incremental processing occurs. This is accomplished through dynamic type checking, and facilitates filter debugging and execution. The graphic filter representations (shown on the screen) are also fully customizable, and can be designed using drawing constructs, or simply loaded as predesigned graphics files.

IV. Conclusion

Flexibility is the key to this utility. Once filters are coded, non-programmer experimenters are able to graphically design and test applicable filter network configurations without requiring additional software support. The development of additional filters further increases the base of available building blocks. The ability of the application to easily integrate filters coded in other languages encourages the filter library to be increased quickly and efficiently. This GUI based tool is very useful, providing researchers greater flexibility in facilitating the design and execution of multispectral experiments.

Our research effort involves the complete implementation of the graphical utility described herein. Several components have already been tested, and additional capability is being added very quickly. Our primary goal is to provide multispectral imagery researchers and analysts a tool that can be used to further guide their own research decisions.