**Implementation of the Bilateral Filter in the LabVIEW environment**

Cristian Ascencio University of Texas at Austin cascensio@utexas.edu Nicholas Dao University of Texas at Austin daonicholas@utexas.edu Vinh Tran University of Texas at Austin vinhtran83@utexas.edu

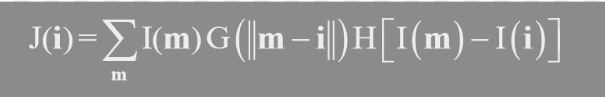
**ABSTRACT**

Our team will present our design process, lessons learned, and the obstacles that we faced when we tried to implement new modules to the existing EE371R (Digital Image Processing class at UT Austin) educational class LabVIEW demonstration. We will be focusing on the bilateral filter implementation. In addition, the image processing that we learned, we will also present our findings about trying to extend the existing class demonstrations. There were many problems that we faced, and although we were successful at fixing some of the problems, our final implementation has bugs, as of the first draft of this report. However, after another few days of debugging, our team finally figured out the source of our errors. Since debugging was such a big process of our design, our team will also present advice, suggestions, and some guidelines for future students who also believe the education VI’s (Virtual Instrument file, LabVIEW programs) are helpful as a visual aid to the lecture material and would like to also take on a similar project.

**1. INTRODUCTION**

Our team originally wanted to take on this project to extend the current set of educational VI demonstrations because of how much the demonstrations have helped us visualize the class material. The demonstrations have helped us get an understanding of how operations affect an image. Originally, we proposed to Dr. Bovik a list of ideas, and he suggested that we try the simpler topics, and move on to more complicated projects if the simpler topics could be done. However, our team faced many issues along the way after designing the bilateral filter in the LabVIEW environment. So, instead of our original plan to implement multiple filters and image processing algorithms, we will be presenting our learnings, design process, and the many obstacles that we faced with the LabVIEW implementation of the bilateral filter in the legacy code environment.

**2. THE DESIGN AND IMPLEMENTATION**

The main design that our team based our bilateral on was from the class lecture slides [1] (Figure 1), with some help from some open-source C++ code for reference [2]. We wanted to start out with the simplest design so that we could worry more about the intricacies of working with legacy code and legacy code environments. We implemented the bilateral algorithm without any shortcuts or any methods to reduce the number of computations. The algorithm to apply the bilateral filter requires 4 loops, where we iterated over the vertical (outer most loop) and horizonal (second outer most loop) axes of the images. We wanted to be able to apply the filter to the entire image, so we need to iterate over all the pixels. The 2 remaining loops are to iterate the vertical (second inner most loop) and horizonal (inner most loop) pixels of the window, so that we are able to compute the new pixel value based on the pixel’s neighbors (which depend on the window shape and size).

**Figure 1**

Our design was implemented in the LabVIEW environment and was based on the homomorphic VI that is provided to us in the class demonstration files. In addition, we wanted our design to work with the “The Essential Guide to Image Processing” (TEGTIP) application, which is given to us as an executable file. To help integrate our new VI with the TEGTIP, we used the homomorphic VI file as a foundation, and changed the inner computations to perform the bilateral filter.

**Figure 2**

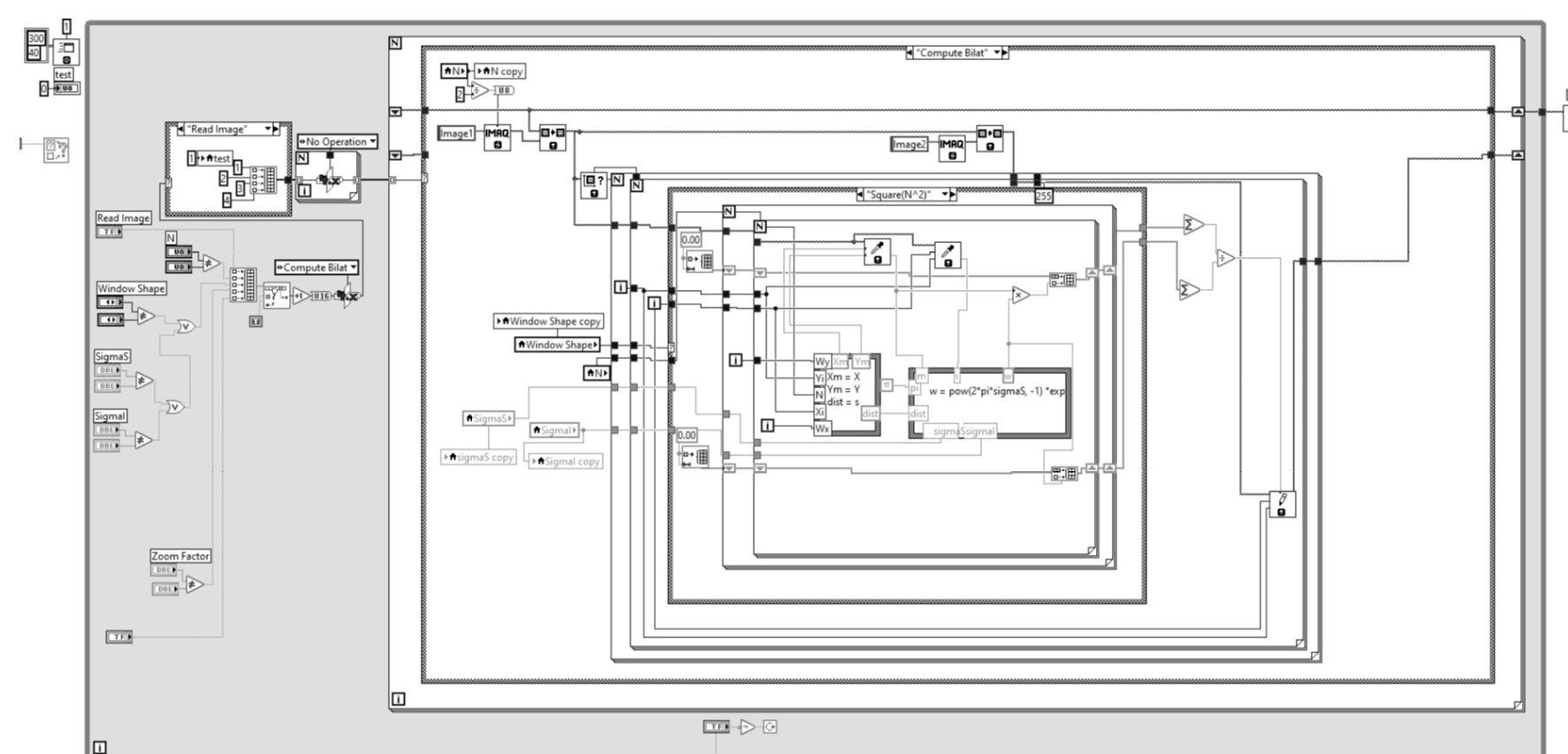
Even though we wanted to be able to have our VI file run straight from the TEGTIP application, we ran into many issues. We tried to contact NI engineers about this issue and Dr. Bovik also tried assisting us with that, but we never got the source files to compile the TEGTIP application, so we were not sure how to resolve our many issues with compatibility. As a workaround to this issue, we settled with building the VI as an executable application, capable of running on a computer with the LabVIEW Run-Time Engine. We will further discuss this issue later in this report, as well as the steps we took to try and prevent errors, and the steps we took to try and resolve these issues.

At the time of writing our first draft of report, our team has the bilateral filter VI working for values of sigma higher than 10, and the algorithm itself is computationally heavy (compared to other algorithms, and for big window sizes) and is computationally inefficient. However, after debugging our errors, our team has implemented the bilateral filter for all values of sigma that fit into the data word size. In addition to fixing our errors regarding the sigma values, we also were not able to get the “cross” and “X” shaped windows working, as of the first draft of this report. However, like with the sigma values, we were able to debug our errors and our final implementation works for all sigma values within the data type range and the window shapes for implementing the bilateral filter work too. In Figure 2 and Figure 3, we have examples of the bilateral filter working with , and for



**Figure 3**

**Figure 4**

In Figure 4, we show our block diagram of the working bilateral filter implementation.

**Figure 4**

Next, we will discuss the problems we had with our first implementation and suggest potential changes to our final implementation.

**3. OBSTACLES FACED**

The implementation of the bilateral filter is simple at a high level, however, we wanted to be able to implement this in the LabVIEW environment so that it could potentially be added to the set of educational VI’s. Implementing this would be much simpler with a programming language like MATLAB, Python, or C++, especially with so many libraries that exist for different applications. So, learning how to implement the bilateral filter in LabVIEW was one of our main challenges, but after reading documentation and understanding how the other educational VI’s were implemented, we could start to try and implement our design.

All our team members have had very minimal experience with LabVIEW prior to this project so creating a file in this environment proved to be very difficult at first. However, once we understood how LabVIEW worked, we were able to implement the algorithm. Another part of the implementation process that proved to be a problem was the debugging process for LabVIEW. Since all our team members are only familiar with debugging lines of a written programming language and not a graphical programming language, it was very hard to find out why there were errors.

**3.1 Team member contributions**

It was all our team members’ responsibilities to learn the basics of LabView, however, each of us helped contribute to different areas of the project.

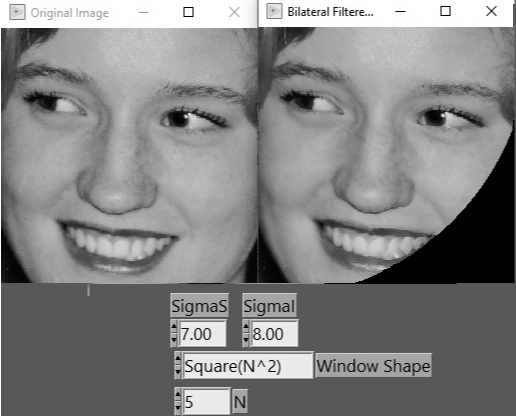
Nicholas was the main communicator between Dr. Bovik and the team, he helped set up the meetings that we had with Dr. Bovik. In addition, Nicholas helped with the appearance of the VI, making sure it looked like the other VI’s. Another way Nicholas helped contribute was helping understand the bilateral filter algorithm to then help explain to the implementor. The algorithm is based upon the slides from class as well as from a C++ open-source file, which contained code for how the algorithm computes the values needed. Along with his other responsibilities, Nicholas helped debug the LabVIEW program after it was implemented in the first few drafts.

Cristian was the main designer of the team, since he implemented the bilateral filter algorithm for the first few drafts of our design and, after debugging and revisions, our final implementation. Cristian also implemented the different algorithms to calculate pixel values for different window sizes and shapes. Cristian was the main implementor of the bilateral filter algorithm in LabVIEW, with Nicholas and Vinh helping by studying the theory/algorithm and debugging.

Along with the helping understanding of the algorithm and the debugging, Vinh also was the main person in-charge of researching the LabVIEW specific blocks that were implemented in the original VI’s, specifically the image processing modules (Vision Development Module) [3]. Since none of us were very comfortable with LabVIEW in the first place, we were not familiar with how the LabVIEW add-ons worked, so Vinh helped us with understanding that, through his research.

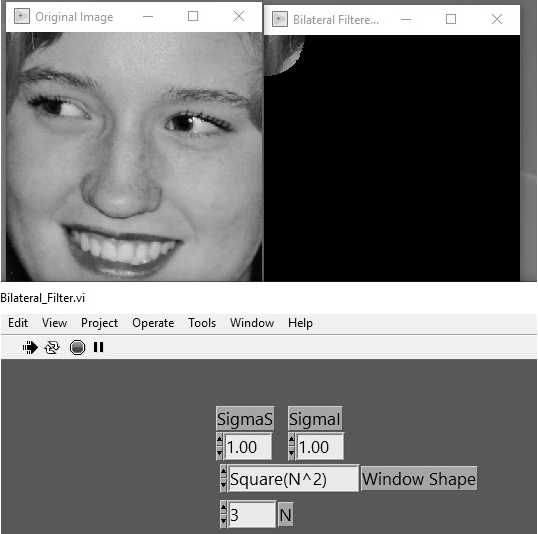
**3.2 Problems with our current design**

**Figure 5**

As mentioned earlier in our “Design and Implementation” section, we mentioned that we could not get the source files for the TEGTIP application and were running into errors when calling our bilateral filter VI from the application. In addition, we did not have an efficient algorithm for calculating the values needed to implement the bilateral filter, so our implementation takes a long time (in comparison to the existing education VI’s) to output the bilateral filtered image.

Earlier we mentioned that there was a problem with the VI when there were values of sigma under 10. After some experimentation with different sigma values, we saw that sigma values until 7 or 8 would not compute the bilateral filtered image correctly, so we decided, as a workaround, to make the minimum allowable sigma have a value of 10. Our team believes that this error is due to a computational underflow, where the values are being truncated down to 0, which would explain the sections of black within the image, but we were unable to successfully debug the error. To further give a better understanding of the problem we were originally facing, we have these errors shown in Figure 5 and Figure 6, with the same sigma values as Figure 2 and Figure 3, respectively.

**Figure 6**



We have just discussed why we believed our first few implementations of the bilateral filter did not work, however, when we debugged our LabVIEW code, we realized the error was attributed to the incorrect calculation of distance between the source pixel and its neighbors.

**4. SUGGESTIONS FOR IMPROVEMENT**

Our team realizes that the design/algorithm that our team implemented is not the most efficient design and there is room for improvements in many areas of our project. Our team wanted to provide suggestions for improvements to our implementation.

The first of our suggestions is that the implementation could have been done with less computations as our naïve implementation is very redundant. Instead, our team could have pre-computed values and then stored them into a look-up table, which would speed up the bilateral filter operation. Our current design iterates over all pixels of the image and through all the pixels of the window that the bilateral filter operation is to be operated with respect to. Due to this, our computation is slow and would benefit if values were pre-computed.

Secondly, we suggest implementing the filter using the Square, Cross, and X window shapes with smarter indexing. Currently, these windows essentially iterate over a NxN window that only does calculations if the given index checks are true. A few suggestions would be to pull up the incrementation of the window’s y-coordinate since it doesn’t need to be refreshed until after the innermost loop iterates through all the x-coordinates. Additionally, it is possible to reduce unnecessary calculations in the Cross and X windows by indexing into directly to the appropriate neighbors instead of scanning the area for them.

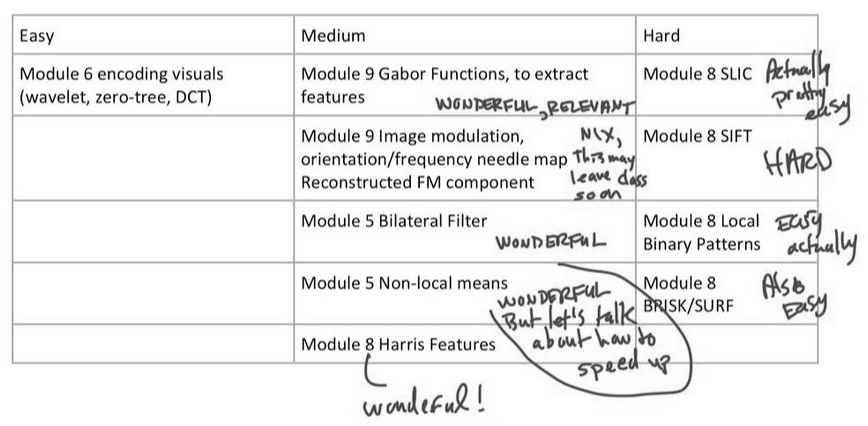
Finally, our team believes that if we were in contact with a National Instruments (NI) engineer earlier or with an NI engineer who knows about the TEGTIP application, we would have been more successful with integrating our VI with the TEGTIP application. We know that Dr. Bovik was helping us with getting into an NI engineer who had expertise with this application, but we also understand that his previous contact no longer works for NI, so that was another complication that was not in our control.

**5. ADVICE TO FUTURE STUDENTS**

To further discuss our final suggestion for improvement, regarding contact with NI, we wanted to dedicate a section to discuss advice to future students who wish to extend the existing set of educational VI demonstrations. This section will essentially go over a few things we wished we knew in advance, which would have made our project go much smoother, including contact with NI.

First, we suggest that students open the source code of the existing VI’s and study that closely, earlier in the semester is better. The previous VI designs work well, and we believe that it makes the implementation process a bit simpler if you work with an already existing block diagram. However, students will not be able to take advantage of the existing VI’s without understanding how they work.

Next, with regards to getting into contact with an NI engineer, our team suggests that a future team gets into contact with Dr. Bovik early about this project, so that the appropriate contact/engineer at NI can be helpful in assisting the team with source code, compilation, and the LabVIEW environment (specifically with respect to the TEGTIP application). We know that unforeseen complications arose during the semester because Dr. Bovik’s contact no longer works at NI, so if a future team could work out all of those details early in the semester, that would make the project much smoother and have a better chance of being implemented with fewer or even no errors.

Finally, our team had some trouble trying to decide which topics we were interested in or wanted to implement, but we knew if we chose earlier in the semester, it would make the project go smoother. The problem, however, is the fact that students will most likely be unfamiliar with the topics until we are presented with them in class, so choosing topics early is difficult. We recommend future students read ahead in the lectures and see which topics sound interesting, as well as doing your own research, then contacting Dr. Bovik about which topics are doable and which are too complicated. To help give the students an idea of what we originally came up with as potential ideas, and for an idea of feedback from Dr. Bovik, we have included a picture of our rankings of topics and our perceived difficultness of implementing the algorithm, and then Dr. Bovik’s thoughts about which are good projects and whether or not the projects are actually easy or hard. ****This picture is displayed under Figure 7.

**Figure 7**

**6. CONCLUSION**

Our team originally wanted to implement multiple topics and image processing algorithms in the LabVIEW environment, to be potentially incorporated into the existing classroom VI’s. Our team has enjoyed and appreciated the demonstrations in class, since they elevate the learning process and provide an additional medium for learning the material. Although our team ran into many complications during the project, we were able to implement our bilateral filter using Row, Column, Square, Cross, and X shaped windows. We have some additional bugs regarding the output image, but these bugs are masked with our implementation, since were never able to successfully pinpoint the source of error, but this was as of the time of our first draft of the report. We have since fixed those bugs and the sigma values are being computed correctly.

In addition, our team was not able to call the VI from the TEGTIP application, but instead, as a workaround, decided to build the VI as its own executable file, which worked fine.

Our team learned a lot about the bilateral filter algorithm as well as the inner workings of other image processing algorithms such as the homomorphic operation as well as the grayscale morphological operation (since we used these to study the inner workings of the VI’s). In addition to the image processing techniques and topics, our team also learned a lot about LabVIEW, working with legacy code, and the complications that come along with doing working with legacy code. We really enjoyed this project and even though our implementation is not fully optimized, it is functional enough to be a basic demonstration of the bilateral filter. We hope that our basic implementation could potentially be improved upon in the future to be more computationally efficient, have fewer or no bugs, and can be appropriately incorporated into the TEGTIP environment.

**7. REFERENCES**

[1] A. Bovik, “Module 5” *In-Class Lecture Notes*. University of Texas at Austin. File Type: PowerPoint [slides: 7-9]

[2] A. C. Aydin, “Bilateral Filter Implementation,” *GitHub*, 26-Nov-2016. [Online]. Available: https://github.com/anlcnydn/bilateral. [Accessed: 01-Nov-2019].

[3] “Image Processing VIs,” *Image Processing VIs - NI Vision 2019 for LabVIEW Help - National Instruments*, Mar-2019. [Online]. Available: http://zone.ni.com/reference/en-XX/help/370281AG-01/TOC57.htm. [Accessed: 31-Oct-2019].