Toward a Scalable Bank of Filters for High Throughput Image Analysis on the Cray Urika-GX System

Shilpika <shilpika@anl.gov>[1], *Nicola* Ferrier <nferrier@anl.gov>[2], and *Venkatram* Vishwanath <venkat@anl.gov>[1]

[1] *Leadership Computing Facility, Argonne National Laboratory, 9700 Cass Ave, Illinois, United States*
[2] *Mathematics and Computer Science, Argonne National Laboratory, 9700 Cass Ave, Illinois, United States*

Abstract. High throughput image analysis is critical for experimental science facilities and enables one to glean timely insights of the various experiments and to better understand the physical phenomena being imaged. We present the design and evaluation of banks of filters, the core building blocks for high throughput image analysis, on the Cray Urika-GX system. We describe our infrastructure developed with Apache Spark. We scaled this to 800 cores of the Urika-GX system for analysis of a Combustion engine dataset imaged at the Advanced Photon Source at Argonne National Laboratory and observe significant speedups. This scalable infrastructure now opens the doors to the application of a wide range of image processing algorithms and filters to the large-scale datasets being imaged at various light sources.

1 Introduction

High throughput image analysis is of paramount importance to experimental sciences facilities such as the Advanced Photon Source (APS) at the Argonne National Laboratory, Advanced Light Source at SLAC Accelerator National Laboratory and the Spallation Neutron Source at the Oak Ridge National Laboratory. This analysis capability would enable one to glean timely insights from the various experiments and to better understand the physical phenomena being imaged. The Cray Urika-GX provides support for streaming of data and the software environment together with high-performance architecture capabilities for analysis of large-scale datasets. We present the design and evaluation of using banks of filters on the Cray Urika-GX system and evaluate the performance with combustion engine datasets imaged at the APS. Analysis of images using banks of filters is a common pre-processing step for image analysis applications and is the basis of many object recognition and machine learning applications. For large image datasets performing convolutions in parallel greatly reduces overall image processing times and permits the use of large filter banks (e.g. AlexNet [1]). Identifying an optimum set of filter banks, with the ability to switch between different types of filters (gabor, edge/bar, and radial filters) could be used for scientific evaluation and analysis in many scientific imaging applications. The dataset sizes currently imaged at the APS are in the range of gigabytes to terabytes, and this is expected to increase in the very near future, to the petabytes range with the upcoming APS upgrade. We expect similar increases in data sizes at other facilities, all requiring scalable high throughput image analysis infrastructure.

Toward realizing this vision, we analyze x-ray video images of fuel spray from different combustion engines imaged with variation in fuel quantity, composition, and nozzle pressure. Of particular interest is the morphology of droplets of fuel within the spray: their size and scale varies along and across the spray pattern. Extracting quantitative data can inform computational models to understand and improve engine performance. The image analysis pipeline is split into three stages: Image normalization, filtering using a bank of filters to extract morphological details and additional processing to extract quantitative data. We use Apache Spark [2], a parallel data processing engine, to implement our bank of filters. The Urika-GX system, with its system architecture features, including Aries interconnect and deep memory hierarchy, and its software environment, provides for efficient and scalable execution of Spark.

In this paper, we present a high throughput image analysis pipeline using a parallel bank of filters on the Sage Urika-GX platform. This infrastructure is expected to be used in a number of image analysis applications with the option of choosing filter banks and further tuning for different filter scales and orientations, allowing for application specific tuning. We scaled this infrastructure to 500 cores of the Urika-GX system for analysis of combustion engines and observe significant speedups. This scalable infrastructure now opens the doors to the application of a wide range of image processing algorithms and filters to the large-scale datasets being imaged at various light sources and to glean timely insights.

The outline of the paper is as follows. In Section 2 filters for extracting features of interest are presented and algorithms used to extract morphological features are discussed. Section 2 also presents some initial results of filtering with feature extraction. Initial scaling results for the bank of filters on the images using the Sage Urika-GX system and the Cooley System are presented in Section 3. We conclude and discuss our next steps in Section 4.

2 Bank of Filters and Detectors

Figure 1 shows the pipeline of our implementation which is split into three stages of image analysis: Image normalization, filtering using a bank of filters to extract morphological details and additional processing to extract quantitative data. In each x-ray image sequence, the first N (10-20) images, with no spray present, are averaged to be used as a background image. For image normalization, we perform a pixel-wise division with the background image in all the subsequent images of the series. The second stage in the analysis is image filtering using filters including Gabor, edge, bar, and radial filters. For our combustion application we filter the images using each of the filter types. The images are filtered by convolution at every pixel with the filters chosen. The result traces the fuel flow, and for certain filters, the filter response correlates to the thickness of the spray. Stage three of processing involves analysis of the filter response. For our application the post-processing involves finding the regions with the maximal filter response and computing fuel droplet morphology via ridge detection [3] (to reduce the regions of interest to a strand of pixel locations while preserving the extent and connectivity of the original regions and discarding most of the original foreground pixels). Next we extract quantitative information such as the length, width, and orientation of the features (fuel spray droplets) for each image in the time series.

2.1 Gabor Filter

Gabor filters are Gaussian functions that are sinusoidally modulated with optimal joint resolution in frequency and space domains [4]. They are bandpass filters that are directionally selective in nature and could be used to extract edges/ridges in images. Gabor multi scale/orientation filter bank reports excellent texture representation and discrimination capabilities [5]. Gabor filters have been thought to simulate the receptive fields of cells in the striate cortex. Two-dimensional Gabor filter is a complex sinusoidally modulated Gaussian function with the response in spatial domain (Eq. (1)) and in spatial-frequency domain (Eq. (2)):

$$f(x, y; \lambda, \theta, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left\{ -\frac{1}{2}\left[\frac{G_1^2}{\sigma_x^2} + \frac{G_2^2}{\sigma_y^2} \right] \right\} \times \exp\left\{ i \, . \, \frac{2\pi G_1}{\lambda} \right\} \tag{1}$$

where

$$G_1 = x \ \cos\theta + y \ \sin\theta,$$
$$G_2 = -\, x \ \sin\theta + y \ \cos\theta$$

and σ_x and σ_y are the standard deviations along x and y directions. λ represents the wavelength of the sinusoidal factor, θ represents the orientation of the normal to the parallel stripes of a Gabor function

$$F(u, v; \lambda, \theta, \sigma_x, \sigma_y) = C \exp\left\{ -2\pi^2 \left(\sigma_x^2 \left(H_1 - \frac{1}{\lambda} \right)^2 + \sigma_y^2 \left(H_2 \right)^2 \right) \right\} \tag{2}$$

where

$$H_1 = u \ \cos\theta + v \ \sin\theta,$$
$$H_2 = -\, u \ \sin\theta + v \ \cos\theta,$$
$$C = constant$$

The parameters in the equation (1) σ_x, σ_y and λ represent the size of the lines and curvilinear structure. $\sigma_y = e\sigma_x$ where e is the elongation of the filter along the orientation set at θ with respect to its thickness. In our bank of filters application, we can adjust the parameters $\sigma_x, \sigma_y, \theta, \lambda$ and also specify the kernel size of the resulting Gabor filter. The real Gabor filters are used to detect elements that are brighter than their background i.e. element s with positive contrast. Suppose $A(x, y)$ is an image being processed and $\theta(x, y)$ is the angle of the fuel spray at pixel location (x, y). If $G_i(x, y)$, where i = 0,1...N, form a bank of Gabor filters such that each filter uniquely represents a parameter set consisting of $\sigma_x, \sigma_y, \theta, \lambda$ and filter size, then the Gabor filtered images are given by $R_i(x, y) = A(x, y) * (G_i(x, y)$ where the asterisk represents convolution operation and i = 0,1...N . Each of the resulting images have an orientation field equivalent to that of the corresponding filter. For larger values of σ and λ thicker lines in the image will give higher responses and the finer lines will be ignored. We combine all the results from the filters to give us the best response at each pixel and the final image, representing the maximal filter response is sent to the next stage of ridge detection.

In the middle of 1990s, some researchers applied Gabor filters to feature extraction for character recognition [6, 7]. These methods achieved some progress for noise tolerance. But considering the cost of computation required, the improvement in performance was trivial. With the parallel bank of filters with each image being processed per node we could achieve significant improvement in computation time.

2.2 Edge/Bar Filter

In this study, we use the Maximum Response-8 (MR-8) filter set [8], shown in Figure 2. The MR-8 filter set is based on the Root Filter Set (RFS) which consists of two anisotropic filters (one edge and one bar filter, at six orientations and three scales) and two rotationally symmetric filters (one Gaussian and one Laplacian of Gaussian). The Bar and the Edge filter in the MR-8 set have six orientations and three scales each, however, we provide support for customizing scales and orientations leading to different number of the filters for processing.

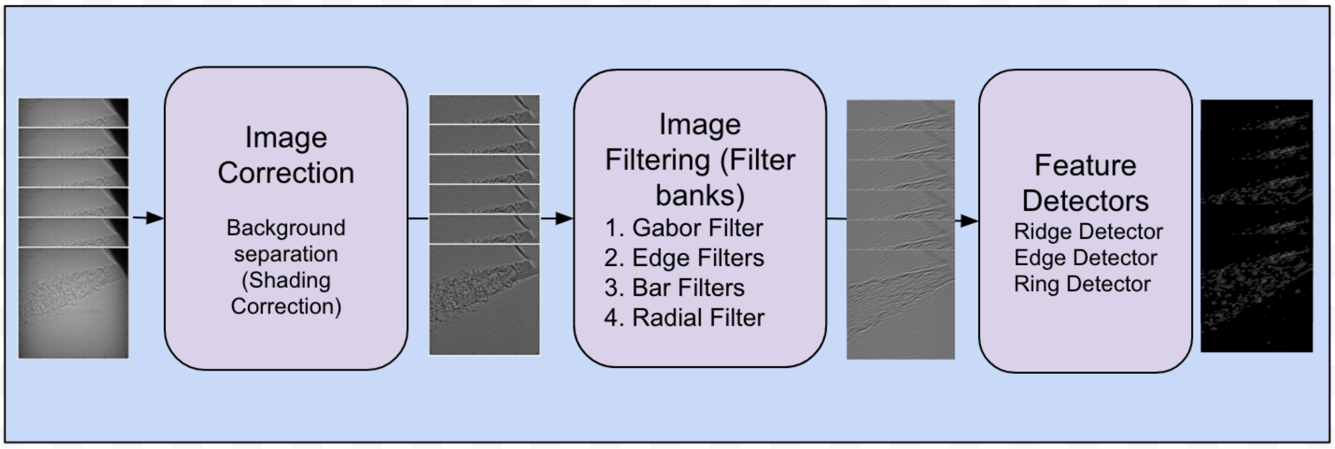

Figure 1. Architecture Overview: The three stages of analysis include Image Correction, to eliminate background noise and for shading correction, Image Filtering, to choose a suitable filter(Gabor, Edge, Bar, and Radial filters) for feature extraction, and Feature Detector, to detect and extract quantitative data of the filtered features

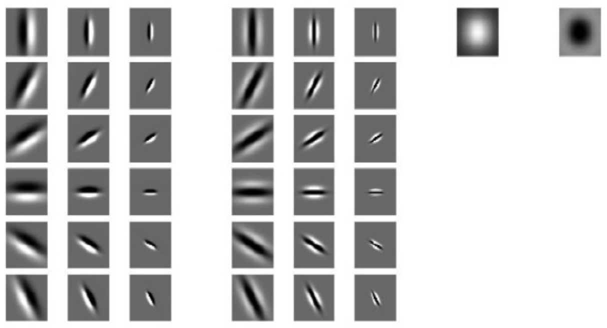

Figure 2. The MR-8 filter bank consists of 2 anisotropic filters (an edge (columns 1-3 above) and a bar(columns 4-6 above) filter, at 6 orientations and 3 scales), and 2 rotationally symmetric (columns 7 and 8 above) ones (a Gaussian and a Laplacian of Gaussian)

Edge filters are used to filter useful structural information about object boundaries. An optimal edge filter implementation results in edges being marked as maxima in gradient magnitude of a Gaussian-smoothed image.The Bar filters are used to extract ridge details on the images. It is useful sometimes to extract rotationally invariant feature and the MR8 filters provide two rotationally invariant filters. Across each scale, the response with the highest magnitude is selected for all orientations of the filter resulting in three filters each for Bar and Edge, these results are then merged across all scales to give the final response. For our study, we chose to use only the Bar and Edge Filters from the MR-8 filter bank with the option to choose the scales and orientations of the filter. For rotationally invariance we use the Radial filters mentioned in 2.3

2.3 Radial Filters

Radial filters are rotationally invariant filters or isotropic "Gabor-like" filters. For the bank of filters, radial filters could be used to highlight features that are symmetric along all orientations. These filters combine frequency and scale [9], as evident in equation (3):

$$F(x, y; \tau, \sigma) = F_0(\tau, \sigma) + \cos\left(\frac{\sqrt{x^2 + y^2}\, \pi\, \tau}{\sigma}\right) \times \exp\left\{-\frac{x^2 + y^2}{2\, \sigma^2}\right\} \tag{3}$$

where τ is the number of cycles of the harmonic function in the Gaussian envelope of the filter, mainly used in the context of Gabor filters. $F_0(\tau, \sigma)$ is added to obtain a zero DC component, this makes the filters robust to illumination changes, as we obtain invariance to intensity translations [9].

As in the case of filters mentioned previously we can customize values for τ and scales.

2.4 Ridge Detection

The results of the second stage of analysis (filter response) are shown in Figure 3. For the combustion application the best results are obtained by using the Gabor filters, in that the resulting image shows patterns that closely trace the fuel flow in the original image. Results for Radial filters show tiny "blobs" being identified and hence for this application this choice of filter would not be suitable. Edge and Bar filters trace the fuel flow fairly closely. In each of the case the results vary by adjusting the filter parameters. For example, we could tune the radial filters to detect larger "blobs" by changing the scale and τ and then filter the images using each of the filters. For this application, the goal is to have a filter response that closely correlates to the thickness of the spray.

The filtered images are processed to extract quantitative data. To detect features like ridges in an image various computationally expensive techniques [10–12] using some kind of unsupervised learning techniques have been used. For our combustion study, we used a ridge detector

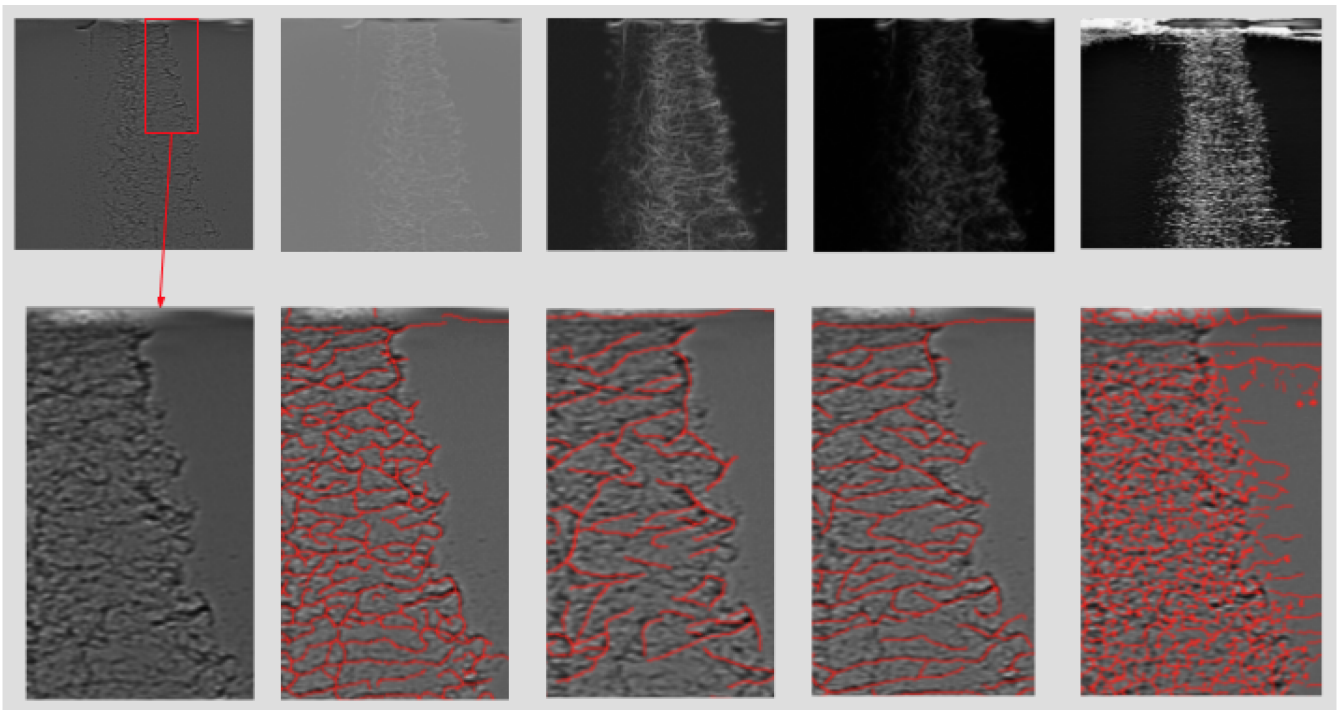

Figure 3. (1)Top leftmost image is the image to be processed, top row images (2),(3),(4),(5) are the results of filtering using Gabor filters with 4 scales and 35 orientations, Edge filters, Bar filters each with 8 scales and 35 orientations and Radial filters with 4 scales and τ between 1 to 4 for each scale

[3] which identifies the fuel spray pattern. The detected ridge location is the central axis of the detected feature (fuel droplet). Here explicit models are used for different line profiles and a scale space analysis on these models was used to develop an algorithm to detect the feature central axis location and the width; both can be extracted with sub-pixel accuracy. A common approach of using derivatives of Gaussian masks to determine the derivative of the image is employed. In this section a brief summary of the ridge detector [3] is provided.

To detect a ridge with a profile matching parabolic (4) or symmetrical bar-shape, given by the function $f(x)$, we detect the points where the $f'(x)$ is 0.

$$f(x) = \begin{cases} h(1-(\frac{x}{w})^2) & |x| < w \\ 0 & |x| > w \end{cases} \tag{4}$$

To extract salient locations we look at the magnitude of $f''(x)$ at points where $f'(x)$ is 0. Bright regions on a dark background will have $f''(x) << 0$ while dark regions on a bright background will have $f''(x) >> 0$ [3]. To estimate the first and second derivatives of f(x) we convolve the image with the derivatives of the Gaussian smoothing kernel. It is the Gaussian kernel that makes the inherently ill-posed problem of estimating the derivatives of a noisy function well-posed [13, 14]. A 2D image $f(x, y)$ containing curvilinear structures can be thought to exhibit 1D ridge profile in the direction perpendicular to the ridge. Let this direction be $p(t)$ for a curve $c(t)$ on a 2D structure where the first directional derivative is 0 and second directional derivative has a large absolute value. The next step would be to determine the direction the ridge takes at each pixel point in the image by computing the partial derivatives dx, dy , dxx, dxy , and dyy of the image by convolving it with the Gaussian kernels shown below [3]:

$$\begin{aligned} G_{x,\sigma}(x, y) &= G_\sigma(y)\, G'_\sigma(x) \\ G_{y,\sigma}(x, y) &= G'_\sigma(y)\, G_\sigma(x) \\ G_{xx,\sigma}(x, y) &= G_\sigma(y)\, G''_\sigma(x) \\ G_{xy,\sigma}(x, y) &= G'_\sigma(y)\, G'_\sigma(x) \\ G_{yy,\sigma}(x, y) &= G''_\sigma(y)\, G_\sigma(x) \end{aligned} \tag{5}$$

Therefore, the direction of the ridge in the image function $f(x, y)$ which is given by the eigenvalues and eigenvectors of the Hessian matrix (6) is the direction in which the second directional derivative has a maximum absolute value which is also the measure of the strength of the ridge.

$$H(x, y) = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{xy} & d_{yy} \end{pmatrix} \tag{6}$$

The computation is done in a stable and efficient way using one Jacobi rotation to remove the d_{xy} term [15]. The direction perpendicular to the ridge given by the eigenvector p_x, p_y which has $\|(p_x, p_y)\| = 1$. A quadratic polynomial is used to determine whether the first directional derivative along (p_x, p_y) is 0 [3]. This point or the subpixel location of the ridge will be given by

$$(X, Y) = (Ap_x, Ap_y) \tag{7}$$

where

$$A = -\frac{d_x p_x + d_y p_y}{d_{xx}p_x^2 \;+\; 2d_{xy}p_xp_y \;+\; d_{yy}p_y^2} \tag{8}$$

To determine the orientation of the normal $p(t)$ to the ridge direction, the first ridge point has the normal marked to lie on the right $(-90°)$ of the ridge direction. For consequent points the two normals lie on either side of the point with an angle difference of $(-180°)$. The orientation at each point is the minimum angular difference between the orientation of the previous and the next point. After extracting the pixel points that lie along the ridges in the image, they need to be connected by choosing the appropriate neighbor. It is assumed that three pixels would be sufficient to give an accurate estimate of the ridge direction locally. For example, for a pixel (x, y), only the points $(x + 1, y - 1), (x + 1, y)$, and $(x + 1, y + 1)$ are examined with pixel orientation $\in [-22.5°, 22.5°]$. A point is chosen to be a part of the ridge based on the distance between the sub-pixel ridge locations and difference in orientation between the two pixel points. If the algorithm approaches a point which is already a part of another ridge this point is tagged as a junction and the ridges containing these junctions are split into two ridges. In a case where the junction points are not marked and the algorithm identifies two directions of a ridge then it chooses one of the two directions and continues until it reaches a point where no neighboring points exist. Upper threshold values are chosen to determine new ridges, where the starting point of the ridges has the second directional derivative above this threshold. The points that lie along the rest of the ridge need to have a second directional derivative of greater than the lower threshold values. A concept similar to hysteresis thresholding [16]. In cases where there are multiple responses in the direction perpendicular to the ridge they are tagged as processed if the orientation is almost equal to that of (x, y), however, these cases are rare. Hence end of the ridge is reached when the neighboring points lie on another ridge or are tagged.

When using the ridge detector it is important to pick semantically meaningful parameter values for σ upper and lower threshold. The value for sigma should be chosen such that $\sigma \geq \frac{w}{\sqrt{3}}$ with w being the width in pixels of the lines to be identified. If the lower and upper threshold values are for example set to 0 and 1 then even the faintest of ridges will be detected.

Next step would be to determine the width of the "ridge" identified previously. The points representing the width lie in the direction of $p(t)$ with each width point lying on either side of the current ridge point and having a maximum absolute gradient value (in image terminology. The ridge detector uses a trivial modification of the Bresenham line drawing algorithm [13] to yield all pixels that this line will intersect [3]. It is only reasonable to search for edges in a restricted neighborhood of the ridge. The width of the ridge is expected to be $\sqrt{3}\sigma$. The edge points are extracted using the facet model which is similar to the approach used for line point detection but with different convolution masks with the smallest mask size 3×3 which is proven to provide accurate results [13]. This approach is computationally inexpensive when compared to a common approach which involves computing coefficients of Taylor polynomials for edge detection [17–19]. There are cases where the edge points are not detected using the facet model due to wide or weak gradients being present next to the lines or at ridge junctions where the width is usually greater than $\sqrt{3}\sigma$. In such cases, the width is computed by interpolation or extrapolation along the orientation of the normals on each point in the line. The width of the ridge is extracted for each ridge point. As we trace along the ridge to determine with, for each missing width on each side of the ridge where widths are available in front and behind the line the current point, the width is estimated by linear interpolation. In a case where the width is missing, for example at the end of a ridge, it will be extrapolated to the last defined width [3].

Using the ridge detector we are able to extract useful quantitative information on each feature identified by the detector. The data extracted includes morphological information such as the spray droplet length, width (average, minimum, maximum), standard deviation, spray start points, centroids and end points etc. Using this information we can perform further studies on how the fuel spray alters over time and as a function of the various parameters (nozzle pressure, fuel composition, etc).

3 Performance Evaluation

We evaluate the performance of the bank of filters on the Sage Urika-GX system and the Cooley cluster at the Argonne National Laboratory.The Sage Urika-GX system is a big data analytics platform from Cray optimized for analytics workflows. The Sage system consists of 32 nodes, consisting of 25 compute nodes and 7 service nodes, which are interconnected using a high-performance Aries interconnect; each node has two 16-core Intel processors, 256 gigabytes of RAM, 800 gigabytes of SSD and 4 terabytes of hard-disk. The nodes have a shared HDFS-based distributed filesystem. The Cooley system is a visualization and analytics cluster consisting of 126 compute nodes. Each compute node has 12 CPU cores with 384 GB of RAM and a NVIDIA Tesla K80 dual-GPU card; The entire Cooley system has a total of 47 terabytes of system RAM and 3 terabytes of GPU RAM. Each node has 345 GB of local scratch space using hard-disks. The system has a GPFS based parallel filesystem. We implemented the the bank of filters using Apache Spark 1.6 and Python 2.7. We also leveraged Jupyter notebooks and this provided support for displaying results on the browser which helped in tuning the parameter set of the filters used. We evaluate the performance of the bank of filters with the combustion engine imaging datasets.

Figure 4 depicts the strong and weak scaling performance on the Sage and Cooley system with Spark for Gabor, Edge, Bar and Radial filters. For strong scaling, we use the bank of filters on 1000 images of the combustion engine dataset. We scale the number of executors(cores) in Spark from 8 to 800 on the the Sage Urika-GX system and the Cooley cluster. The top row of Figure 4 depicts the performance for the Gabor, Edge, Bar and Radial filters on these two systems. As mentioned before, we have three stages in the pipeline, namely the pre-processing stage

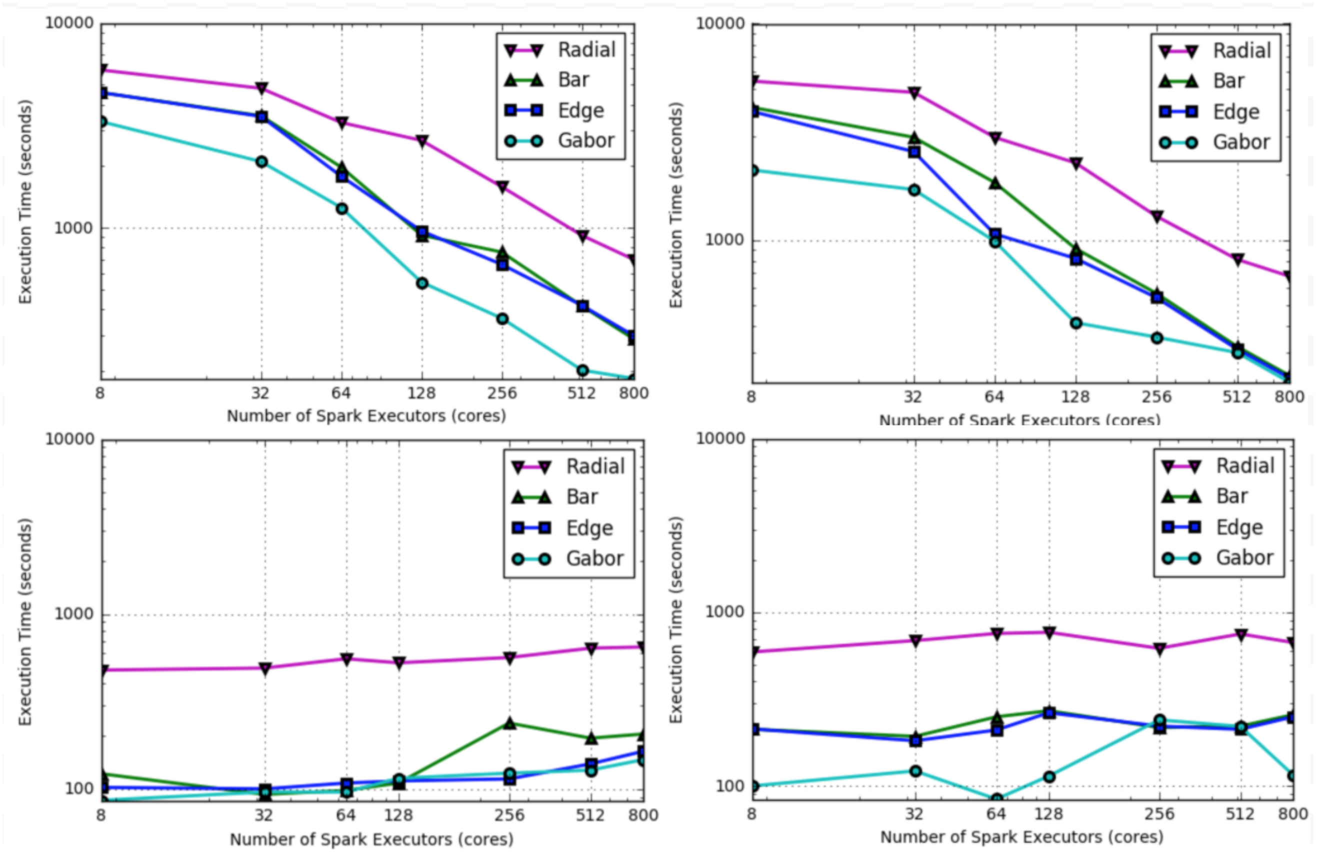

Figure 4. Top row shows the scaling results for strong scaling using 1000 images on the Cray Urika-GX Sage System(1)(left) and Cooley System(2)(right) at Argonne National Laboratory. The bottom row shows the weak scaling results on the Sage Urika-Gx System(3)(left) and the Cooley System(4)(right)

consisting of Image normalization, filtering using a bank of filters to extract morphological details and additional processing to extract quantitative data. The first stage consists of reading the images and necessary communication for the normalization. The second stage is embarrassingly parallel wherein the execution times of the filters differ primarily due to the filter parameters such as the scales and orientations. The last phase also know as the detection phase is dependent on the number of features identified in the second phase. It also involves writing of the results out to the filesystem. As we strong scale from 8 cores to 800 cores on both the Sage system and Urika-GX, we notice that our execution time decreases. In case of Gabor on the Sage system, our execution time decreases from 3305 secs to 184 secs resulting in a speed-up of 17. For the Cooley system, our execution time decreases from 2105 secs to 202 secs resulting in a speed-up of approximately 10. The primary difference in the execution time between the filters can attributed to the filter parameters. Radial filters are more computationally intensive than Gabor filters and as we increase the number of parameters needed to tune the filter, it results in an increase in the overall execution time.

Next, we evaluate the weak scaling performance for the bank of filters on the Sage and Cooley systems. The bottom row of Figure 4 depicts the weak scaling performance as we scale from 8 images on 8 cores to 800 images on 800 cores. We notice that our bank of filters scale well. As we increase the number of cores, we notice and increase in the total execution time. A part of the reason can be attributed to the either the pre-processing of images that necessitates shuffling or at the end of the computation the results are stored in csv files either on the HDFS storage in Sage or the GPFS in Cooley. Spark provides a mechanism to save files to the file system based on the RDD (Resilient Distributed Datasets) partitions. Although this is a useful feature, for larger number of images(4000-10000 images) saving this data via Spark adds overhead in comparison to saving the files directly to the GPFS or HDFS at the end of processing on each executor core. For example, while processing 4000 images on 500 cores on the Sage system it took 17 minutes to process with saving files directly and took around 20 minutes to process for a collect and save operation via Spark i.e the operation was almost 1.2 times slower for the second case.

In terms of quality, figure 3(b) to 3(e) compare results obtained from using different types of filters. The red overlay is the maximal filter response on the normalized image. For this case study, the Gabor filter bank represents the structure of the spray more accurately as it manages to identify local orientation patterns in the spray. Radial filters are isotropic in nature and are not the appropriate choice for ridge detection. Edge and Bar filters trace the edges and ridges of the spray as shown in figure 3(c) and figure 3(d) respectively.

4 Conclusion

In conclusion, we present a high throughput image analysis pipeline using a parallel bank of filters on the Sage Urika-GX and the Cooley cluster. This infrastructure is expected to be used in a number of image analysis applications with the option of choosing filter banks and further tuning for different filter scales and orientations, allowing for application specific tuning. We scaled this infrastructure to 800 cores of the Urika-GX system and the Cooley system for analysis of Combustion engines and observe significant speedups. This scalable infrastructure now opens the doors to the application of a wide range of image processing algorithms and filters to the large-scale datasets being imaged at various light sources and to glean timely insights.

As a part of our future effort, we plan to extend our bank of filters to include more choices of filters. This will provide the flexibility to use a filter based on the image dataset being processed. For example, for extracting circular structures, use of radial filters would be a more suitable choice. The filter responses can be used as input for various machine learning processing pipelines.

Acknowledgments

This research has been funded in part and used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract no. DE-AC02-06CH11357. This work was supported in part by the DOE Office of Science, Advanced Scientific Computing Research, under award number 57L38, 57K50 and 57L32. We would like to acknowledge Jin Wang, Zhilong Li and Yao Ga for the combustion engine datasets and related discussions.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, in *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Curran Associates, Inc., 2012), pp. 1097–1105

[2] N. Chaimov, A. Malony, S. Canon, C. Iancu, K.Z. Ibrahim, J. Srinivasan, *Scaling Spark on HPC Systems*, in *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing* (ACM, New York, NY, USA, 2016), HPDC '16, pp. 97–110, ISBN 978-1-4503-4314-5

[3] C. Steger, IEEE Trans. Pattern Anal. Mach. Intell. **20**, 113 (1998)

[4] G. D., Journal of the Institute of Electrical Engineers p. 429–457 (1946)

[5] T. Leung, J. Malik, Int. J. Comput. Vision **43**, 29 (2001)

[6] X. Wang, X. Ding, C. Liu, Pattern Recogn. **38**, 369 (2005)

[7] D. Shi, R.I. Damper, S.R. Gunn, **2**, 27 (2003)

[8] J.M. Geusebroek, A.W. Smeulders, J. van de Weijer, Trans. Img. Proc. **12**, 938 (2003)

[9] C. Schmid, *Constructing models for content-based image retrieval*, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (2001), Vol. 2, pp. II–39–II–45 vol.2, ISSN 1063-6919

[10] D. Geman, B. Jedynak, IEEE Transactions on Pattern Analysis and Machine Intelligence **18**, 1 (1996)

[11] J. Wang, J. Qian, R. Ma, *Urban road information extraction from high resolution remotely sensed image based on semantic model*, in *2013 21st International Conference on Geoinformatics* (2013), pp. 1–5, ISSN 2161-024X

[12] M.A. Fischler, H.C. Wolf, *Machine Perception of Linear Structure*, in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1983), IJCAI'83, pp. 1010–1013

[13] D.F. Rogers, *Procedural Elements for Computer Graphics* (McGraw-Hill Book Company, New York, NY, USA, 1985)

[14] L.M.J. Florack, B.M. ter Haar Romeny, J.J. Koenderink, M.A. Viergever, Image Vision Comput. **10**, 376 (1992)

[15] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, New York, NY, USA, 2007), ISBN 0521880688, 9780521880688

[16] J. Canny, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-8**, 679 (1986)

[17] L. Wang, T. Pavlidis, IEEE Trans. Pattern Anal. Mach. Intell. **15**, 1053 (1993)

[18] L. Wang, T. Pavlidis, CVGIP: Image Understanding **58**, 352 (1993)

[19] R.M. Haralick, L.T. Watson, T.J. Laffey, The International Journal of Robotics Research **2**, 50 (1983)