LLNL-PROC-462149

Effective and efficient optics inspection approach using machine learning algorithms

G. Abdulla, L. Kegelmeyer, Z. Liao, W. Carr

November 15, 2010

SPIE Laser Damage Conference
Boulder, CO, United States
September 26, 2010 through September 29, 2010

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Effective and efficient optics inspection approach using machine learning algorithms

Ghaleb M. Abdulla, Laura Mascio Kegelmeyer, Zhi M. Liao, Wren Carr

Lawrence Livermore National Laboratory

Abstract

The Final Optics Damage Inspection (FODI) system automatically acquires and utilizes the Optics Inspection (OI) system to analyze images of the final optics at the National Ignition Facility (NIF). During each inspection cycle up to 1000 images acquired by FODI are examined by OI to identify and track damage sites on the optics. The process of tracking growing damage sites on the surface of an optic can be made more effective by identifying and removing signals associated with debris or reflections. The manual process to filter these false sites is daunting and time consuming. In this paper we discuss the use of machine learning tools and data mining techniques to help with this task. We describe the process to prepare a data set that can be used for training and identifying hardware reflections in the image data. In order to collect training data, the images are first automatically acquired and analyzed with existing software and then relevant features such as spatial, physical and luminosity measures are extracted for each site. A subset of these sites is "truthed" or manually assigned a class to create training data. A supervised classification algorithm is used to test if the features can predict the class membership of new sites. A suite of self-configuring machine learning tools called "Avatar Tools" is applied to classify all sites. To verify, we used 10-fold cross correlation and found the accuracy was above 99%. This substantially reduces the number of false alarms that would otherwise be sent for more extensive investigation.

Introduction

The Final Optics Damage Inspection (FODI) camera system is inserted in the center of the NIF target chamber after a laser shot to acquire images of any or all of the final optics for all 192 beamlines [1]. The acquired images are processed using custom image processing & analysis software[2,3], referred to as the Optics Inspection (OI) package, to detect anomalies on the surface of the optics, with the intention of identifying and tracking laser-induced damage. This software then computes a number of measurements for each site, such as area, diameter, intensity, edge strength, aspect ratio, etc. These measurements are then used to provide information on the condition of each optic to the operators of NIF. This task is complicated by artifacts which are difficult to distinguish from real damage sites.

Figure 1 illustrates the difficulty caused by one particular type of artifact, namely light reflected off the equipment used to support the beamline optics, which we will refer to as a hardware refection. Any reflections identified as such can be discarded preventing wasted time and effort on the intensive investigations and calculations that are applied to sites suspected of being damage sites. In the past such reflections have been identified by a person looking at each image. This process is both tedious and extremely time consuming. In this work we use data mining and supervised machine learning techniques to reexamine the data previously compiled by the OI and discard signals due to hardware reflections.

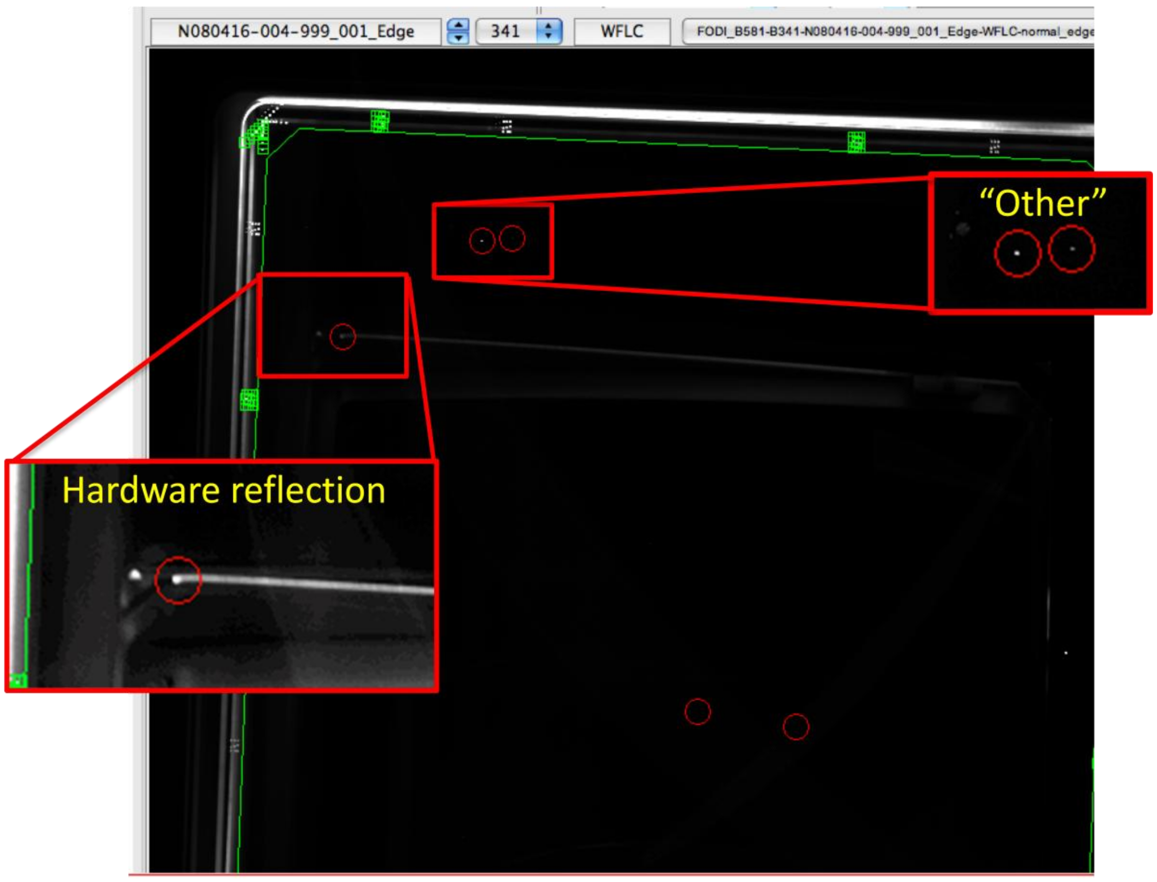

Figure 1: shows an example of a FODI image. Possible damage sites are the small dots identified by a red circle around them. The bright dot at the end of a thin line is a reflection from nearby hardware that was falsely identified by the image analysis software as a possible damage site. The purpose of this study is to automatically distinguish the two by applying machine learning algorithms.

Background on Data Mining & Machine Learning

Data mining is defined as the process of extracting hidden, previously unknown, and potentially useful information, knowledge, or patterns from data. Specifically in [4] Fayyad et al. defines Knowledge Discovery in Databases (KDD) as the process of discovering useful knowledge from data; and data mining is applying the algorithms to extract the knowledge from data [5]. In this paper we don't differentiate between the two activities. We spend a good time interacting with the database to extract the relevant data and ask the correct questions relevant to the practical use case.

A simple definition of learning is "things learn when they change their behavior in a way that makes them perform better in the future" [5]. A machine can learn by observing examples of behavior (e.g. training data) and creating a model for the observed behavior in memory. When a new instance is observed, the machine can use the stored model to give a label or classify this instance. This is why building a model depends heavily on training data that hopefully contains certain patterns with respect to the feature values of the data. For example, in this work we utilize sites previously identified as reflections by humans to compile a training data set. And then use classification algorithms to find a set of features which are likely to be associated with sites identified by humans as hardware reflections. We can then use the same classifier to identify additional reflections among sites not inspected by humans by looking at the relevant features. The sites we will be considering will be classified by looking at the preponderance of the features, not individual features.

Building a model from training data

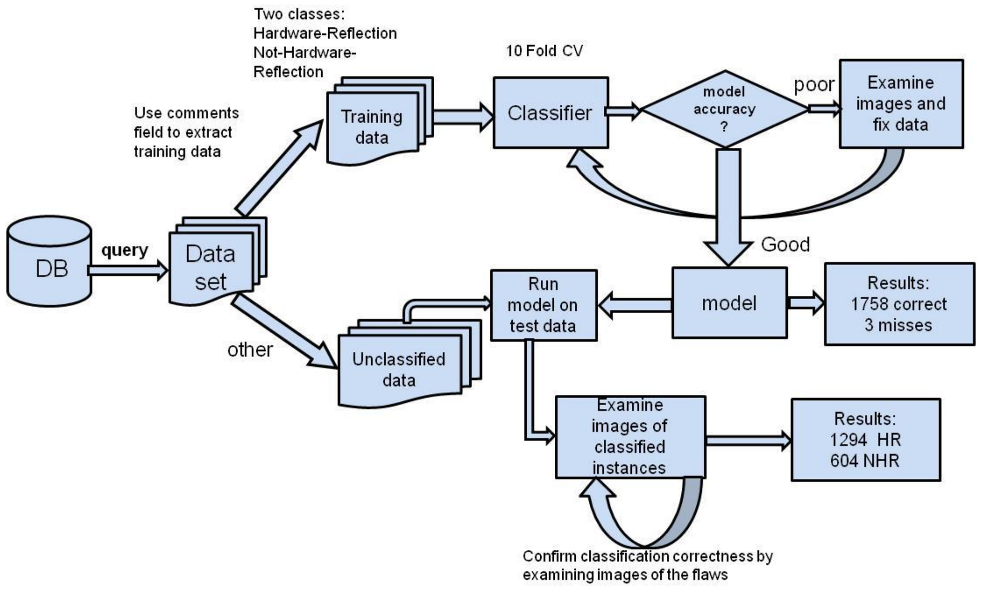
Two classes:
Hardware-Reflection
Not-Hardware-Reflection
10 Fold CV
Use comments field to extract training data
Training data
Classifier
model accuracy ?
poor
Examine images and fix data
DB
query
Data set
Good
Run model on test data
model
Results: 1758 correct 3 misses
other
Unclassified data
Examine images of classified instances
Results: 1294 HR 604 NHR
Confirm classification correctness by examining images of the flaws

Figure 2: The process to prepare a model for data classification

Figure 2 shows the steps that we followed in order to prepare the training data set and choose the classifier. The process includes retrieving data from the database, creating an initial training set, running verification tests, iterating over the training data, validating it with new instances and finally compiling a pristine data set that can be used to generate a model and classify new instances. In the following sections we will describe the process in details.

Mining the database for truth data[1]

The first step was to retrieve, from the database, potential damage sites and their classifications as defined by the human experts. Since multiple experts contributed labels for the data in a relatively unconstrained way, no standardized categories were used to describe the sites they examined.

Our first task was to group the user entered comments into three standard classes: hardware reflection, not a reflection, and unknown. For example, users designated various sites as a "Reflected Flaw", "On Another Surface", or as a "Defect" all of which are positive identifications as some feature other than a hardware refection and included as the category "Not a Reflection". Conversely a number of other monikers could be interpreted as positively identifying a site as a "hardware refection", putting the site in that category. In cases where the comments were not decisive we classified the site into the third, unknown, category. Figure 3 shows an image of each class to illustrate the difficulty in distinguishing between them.

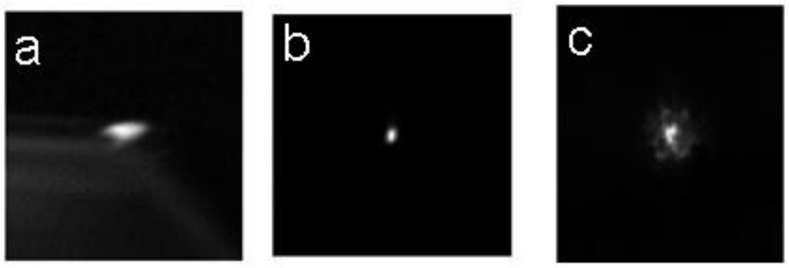

Figure 3: Examples of sites classified as: a) hardware refection, b) defect, not hardware refection, and c) unknown.

[1] We call the data with the correct classes assigned "the truth data"

Retrieving the data from the database, and collapsing the myriad of comments into three classes, though mundane, is the most important step. Indeed first determining appropriate classes and then mapping the available data into them is typically the rate limiting step in the use of a machine learning classifier. Once we have the data with the correct classes assigned, we can pick from several classification algorithms using several publicly available machine learning tools such as Avatar [6] or Weka [7]. The eventual number of instances from the database classified into each category is 1198 hardware reflections, 563 non-reflections, and 1898 unknown. Because the optics are imaged repeatedly over time, a particular site can be represented more than once, that is why sometimes we refer to a site as an instance.

The techniques by which each individual site is identified from the FODI image are beyond the scope of this work and described elsewhere [2]. The function of the automated OI Analysis is to detect candidate sites and extract various attributes or features from the FODI images. Table 1 lists some of the features that OI automatically calculates from a background-subtracted version of the original image. Some features are global, such as where the site is located on the optic, while others are local and use only information from pixels near the location.

Table 1: The attributes OI extracts from the FODI images. All values refer to pixels in or near the extent of the detected site and are measured on a background-subtracted version of the original image.

Area in pixels	Angle of long axis	Standard deviation of pixel values
X location on optic	Short axis of best fit ellipse	Background Mean
Y location on optic	Sum of all pixel intensity values	Maximum pixel intensity value
Long axis of best fit ellipse	Mean intensity	Minimum pixel intensity value

While some of the measurements shown may seem to be less useful than others, it turns out these shown, and others not shown, all contribute in some way and it does not hurt to include any and all of them. Overfitting a model is not a problem for the “ensemble of decision trees” classifier used in this study and irrelevant attributes are simply ignored. For example, salient features (measurements) that might distinguish a red apple from a baseball and a fire truck would almost certainly include color and size, but these are not sufficient. There is too much overlap of these features among these objects. If we add more diversified features, we can extract subtler characteristics and improve our ability to distinguish them. For this simple case, we could add features such as shape (perhaps measured as aspect ratio, circularity or ellipticity), how the size compares to the 3-inch diameter of a regulation baseball (“yes”, ”no”, and other semantic features are allowed), texture measures, how many corners or vertices are included in the shape and so on. The more features, the more refined the separations between these and other, similar objects may become. The use of many features becomes even more important for separating three classes of objects rather than three unique objects because the overlap in any feature may increase substantially due to variations within the individual class. E.g. if toy fire trucks are part of the fire truck class the overlap in the size feature changes significantly. When the classes that need to be distinguished are as similar as those investigated here (hardware reflections and not reflections) a large number distinguishing features becomes critical.

Testing the data with a machine learning classifier

Once a sufficient number of features have been indentified (often an iterative process) and quantified, we then need to test the data using a classifier. A classifier is simply a set of rules which codifies what combination of feature values best identifies individual classes. Luckily there are several available classification algorithms that can be used off the shelf. We first used Weka [7] because of its user interface that includes several other visualization tools which can be used to plot and review results. For production we used Avatar for its performance since it is a C/C++ based tool and is already integrated into OI.

The explorer module in Weka allows a user to choose from several classifiers to test the data. We used Weka to examine the contribution of each feature listed in table 1 for relevance. The most relevant features included the intensity of the signal with respect to the background followed closely by the global location of site and the signal to noise surrounding the site. Using the top four features allowed the classifier to reproduce the opinion of the human experts with 96% accuracy, by including all the features and using the “ensemble of decision trees” available only in Avatar [6], the accuracy rose to 99.8% (see table 2)

Table 2: Classifier results

A C T U A L	**CLASSIFIED AS**		
		Hardware Reflections	**Not Hardware Reflections**
	Hardware Reflection	**855**	2
	Not Hardware Reflection	1	**903**
	Correctly classified Instances: 1758/1761 99.8%		
	Incorrectly classified instances: 3/1761 0.17%		

The matrix with the four numbers in table 2 is called the confusion matrix. It shows how many instances in the original data belong to each class and how many were predicted correctly. The bold numbers (diagonal numbers) are the numbers of instances that were predicted correctly. For example, there were 855 instances that were classified as Not-Hardware-Reflection and two were classified as Hardware-Reflection. The 10 fold Cross Validation test splits the data into two separate data sets one for learning; normally 2/3 of the original data is used for training and building a model while the rest is used to test the ability of the model to predict the classes correctly. The process is repeated 10 times and the results are reported as an average of the 10 tests.

Applying the classifier to the unlabeled data

The third group in figure 3 (the unknown class), are the instances which could not be given a label because the comments were vague or the human experts could not make a determination. When we applied the classifier developed with the use of the truth data, we were able to make determinations for the previously unknown class. We assigned all of the sites from the unknown class to either hardware reflections or not hardware reflections (see figure 4). Avatar was used in this step to grow an ensemble of decision trees using the following options: --use-stopping-algorithm (use out-of-bag validation to automatically determine optimal number of decision trees to use for voting)

--bagging (select from original dataset by sampling with replacement)

--split-method=HELLINGER (Use a splitting algorithm that is more impervious than InfoGain to having under-sampled classes)

Number of instances	Classified as
604	Not-Hardware-Reflection
1294	Hardware-Reflection

Figure 4: The 1898 previously unknown sites were classified as Hardware reflections or not using the same classifier which had a 99.8% accuracy when the answer was known

Discussion and future work

In this paper we show that the use of machine learning to classify and filter hardware reflections is an effective approach, which achieved 99.8% accuracy. These positive results depended first on the availability of feature data that includes the metadata and measurements for each defect candidate and on collecting expert truth for that data. Lastly, it required that we iterate on the steps of refining the truth data. Although the comments entered by the experts varied greatly, we collapsed them to a small number of classifications and used them as truth data. This data set was used with the ensemble of decision tree classifier to predict the categories of the "truthed" sites as well as additional sites that had not been inspected by human experts. These new results were evaluated and when possible, added to the training set and the process was repeated.

The successful results of this work, culminating in the generation of an ensemble of decision trees classifier, has been incorporated into the daily operations for NIF Optics Inspection Analysis and we will continue to apply machine learning techniques to refine, characterize and interpret the data stream coming from the FODI and OI Analysis systems.

Acknowledgements

We would like to thank Ken Manes and Pam Whitman for providing expert truth and painstakingly labeling many hundreds of detections. Their valuable effort made it possible to create the truth data set used in this work.

References

[1] Final optics damage inspection (FODI) for the National Ignition Facility, Alan Conder, Jim Chang, Laura Kegelmeyer, Mary Spaeth, and Pam Whitman, Proc. SPIE 7797, 77970P (2010).[2] Kegelmeyer, L. M., Fong, P., Glenn, S. M., and Liebman, J., 2007, "Local Area Signal-to-Noise Ratio (LASNR) algorithm for Image Segmentation," Proc. SPIE 6696, 66962H (2007).

[3] Kegelmeyer, LM.   Machine Learning to Reduce False Alarms in Shotcycle Inspections , presented at the LLNL CASIS workshop (November 16, 2007).

[4] U. M. Fayyad, G. Piateskky-Shapiro, and P. Smyth, “From Data Mining to Knowledge Discovery: An Overview,” Advances in Knowledge Discovery and Data Mining. Menlo Park, CA. AAAI Press/MIT Press, 1996.

[5] Witten, Ian H., and Eibe Frank. 2000. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. San Diego, CA: Morgan Kaufmann.

[6] http://morden.csee.usf.edu/avatar/

[7] http://www.cs.waikato.ac.nz/ml/weka/

[8] Kegelmeyer, LM. Applying Avatar Machine Learning to NIF Optics Inspection Analysis. Presented to Conference on Intelligent Data Understanding (October 6, 2010) https://c3.ndc.nasa.gov/dashlink/resources/221/ (p. 42)