Similarity Analysis of Supersonic Jet Images

John Hogden, Patricia Fasel, Richard Fortson,
Patrick Kelly, James Howse, & Richard Strelitz
CCS-3

1. Introduction

Our goal is to extend the quantitative image comparison work done by Foster et al. (2002) on experimental and simulated images of a supersonic jet interacting with a counter-propagating shock wave. The 4 ns experimental image is shown in Figure 1 to highlight the structures discussed below. The highlighted structures include the aluminum flat, the two bow shocks, and the jet. The entire set of data we had available consisted of 61 separate images taken at time steps of 0.1 ns from 30ns – 90ns for the simulation and three experimental images.

To compare the experimental image to the simulated image for the same time, Foster et al. compared contours that captured transition points in the experimental image (e.g., the front of the bow shock, the edge of the jet, …) to the contours at the same transmission values for the simulated image. The contours for the simulated image were reported to be within the 12 μm spatial resolution of the experiment at both 4ns and 6ns after the onset of the radiation drive. Foster et al. also examined experimental and simulated images of a shock wave without an interacting jet, and did comparisons of the position of the shock front along the midline.

For a variety of reasons, it would be useful to be able to do such image comparisons automatically. For example, the software used to create the simulated images is periodically changed to incorporate new features. Automatic image comparison could simplify the process of verifying and validating the software changes. Furthermore, it is often necessary to run simulations using a variety of input parameters to determine the best fit to the experimental data. Automatic image comparison would speed up the parameter selection process.

There are some simple approaches to image comparison that are applicable in many situations. As mentioned above, Foster et al. compared the positions of important contours and found the contours to be within error in some regions of the images. This is a quantitative and useful approach to image comparison for these images. However, Foster et al. did not check every contour in every region so might have missed some differences between images. Perhaps an automated procedure should be used to compare every contour throughout the images. Alternatively, since the simulations and experimental images are absolute in time, space, and transmission value, perhaps the images should be compared by taking the root mean squared (RMS) difference between transmission values at each pixel location.

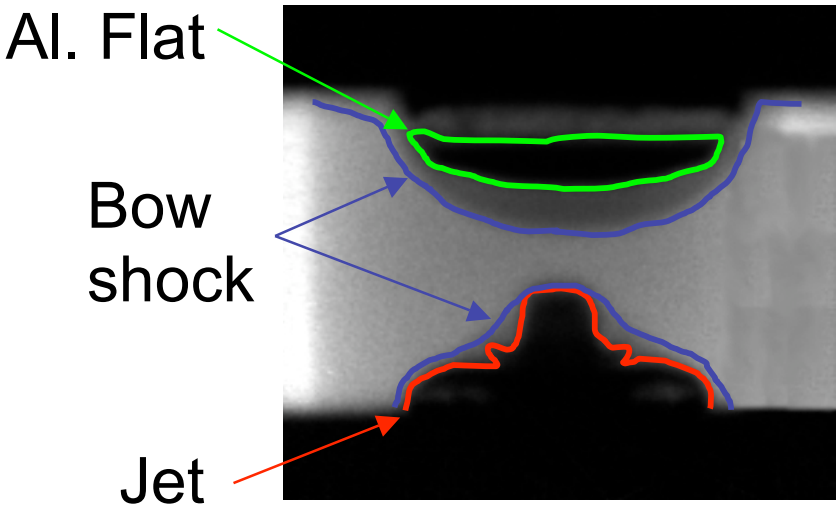

Figure 1. The experimental image at 4 ns. Notable structures apparent to the human eye are the jet, the bow shocks, and the aluminum flat.

While the simple approaches mentioned above could be used on this data, it is not expected that the same techniques will be applicable to future data. For example, while we might always expect to see a jet, small imperfections in the aluminum pin or the laser drive may make the jet bend instead of traveling straight forward, as illustrated in Figure 2. In such a case, we need to use a more abstract notion of the shape of the jet because neither the contour positions nor the RMS difference between transmission values would capture the fundamental fact that a jet exists, has a generally mushroom-like shape, and is a particular size. In cases like Figure 2, comparison of the length of the "stalk" or the width of the "mushroom cap" may be more appropriate. Of course, much more complicated comparisons may be demanded by theory, e.g., if we had a strong theoretical reason to believe that the length of the stalk would change due to slight experimental imperfections that were beyond our control, but that the length of the stalk should always be inversely proportional to the integral of the curvature of the midline of the stalk, then a analysis of the length/curvature ratio would be justified. Unfortunately, we are not aware of any applicable theory.

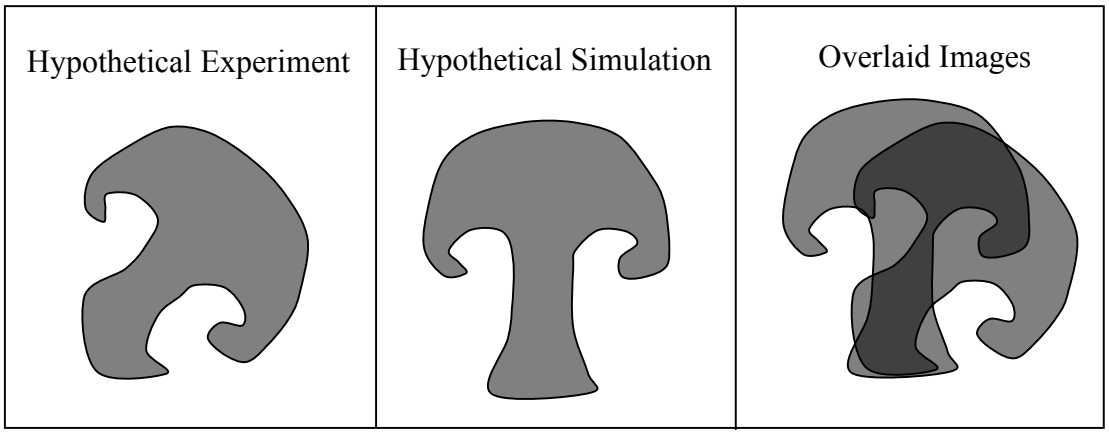

Figure 2. Small changes in the experimental conditions may create a warped jet, as in this hypothetical experiment. Nonetheless, there are clearly some similarities between the experiment and the simulation, despite the fact that neither the contour positions nor the pixel-by-pixel transmission value differences capture the similarity.

In a previous attempt to compare the Foster et al. images using a more general shape metric, the images were first decomposed into Zernicke moments (Cannon et al., 2004). Euclidean distances between the first few Zernicke moments were then used as dissimilarity metrics. However, the results of that experiment are difficult to interpret. For example, when a difference between the experiment and simulation is found, is the difference due to a change in the shape of the bow shock, the shape of the jet, the closing field of view, or the modeling of the gold washer? Knowing the cause of the difference is important because some differences are more important than others. Since a defect in modeling the gold washer is considered less important than a defect in modeling the aluminum jet, we want to weight dissimilarities differently depending on the source of the difference. Furthermore, if we want to use the results to improve the simulations, it is important to know what is causing the differences. Being able to relate differences to simulation input parameters, instead of simulation outputs such as jet shape, could be even more valuable.

In the work presented here, we try to improve on earlier work by separating out differences due to different features in the images. While we could find differences at each contour level, we opted to try to find differences between image structures that are readily apparent to humans, such as the bow shock and jet. Doing so requires the ability to automatically extract structures from the images, which is known to be a difficult task, but an interesting research area. We make simplifying assumptions to allow us to perform object extraction with limited, but adequate accuracy, and focus only on one of the simpler structures to extract, but we believe the approach we have taken can be easily adapted to other problems. The approach amounts to automatically finding a threshold such that the thresholded image meets certain a priori shape requirements. More information about our approach to extracting structures can be found in Section 3.

Zernike moment coefficients are similar to Fourier coefficients in that modeling abrupt changes (i.e. edges) requires many coefficients. We decided not to use Zernike moments on the thresholded images, in part because thresholded images have abrupt changes. Instead, we use the Euclidean distance between Line Scan Transforms (LSTs) of images as our metric (Warnock & Cannon, 2004). The LST is an intuitively appealing approach to comparing shapes of objects in that it is similar in spirit to comparing the positions of structure boundaries. As discussed in more detail in Section 2, to calculate an LST we randomly place lines over an image and measure distances between the points at which the lines enter and leave objects. These distances are accumulated in a type of histogram that is the LST. Note that the LST histogram contains information about the extent of mixing of two materials —big blobs of materials will have large internal line lengths whereas the long, curly, thin bands of materials that characterize more mixed materials will have shorter line lengths. We speculate that the LST gives us information

that is more robust than contour distances and RMSE for the types of deformations shown in Figure 2, although that intuition has not yet been rigorously demonstrated.

Previous unpublished work with the LST on the Foster et al. images showed some severe problems. The blue curve in Figure 3 shows the previously calculated LST dissimilarity between the 4 ns experimental image and each of 50 simulated images (the x-axis label in the plot is inaccurate — the numbers on the x-axis are actually tenths of nanoseconds). Since the simulated images show the time sequence of jet and shock evolution, we would expect the dissimilarity to drop to a minimum near the simulation that is intended to approximate the 4 ns image and would rise as we move away from the minimum. Unfortunately, that is not what we see. Instead, the curve has its minimum near 6.0 ns and has many local minima. Note, however, that all the comparisons were made using a single transmission level of 0.4. Since the transmission levels of the main structures change over time, different structures were being compared over time. In the work we report here, different thresholds are used over time in an attempt to track the interesting structures. As seen in Section 4, the new method of transmission selection dramatically alleviated the problems seen in the previous LST work.

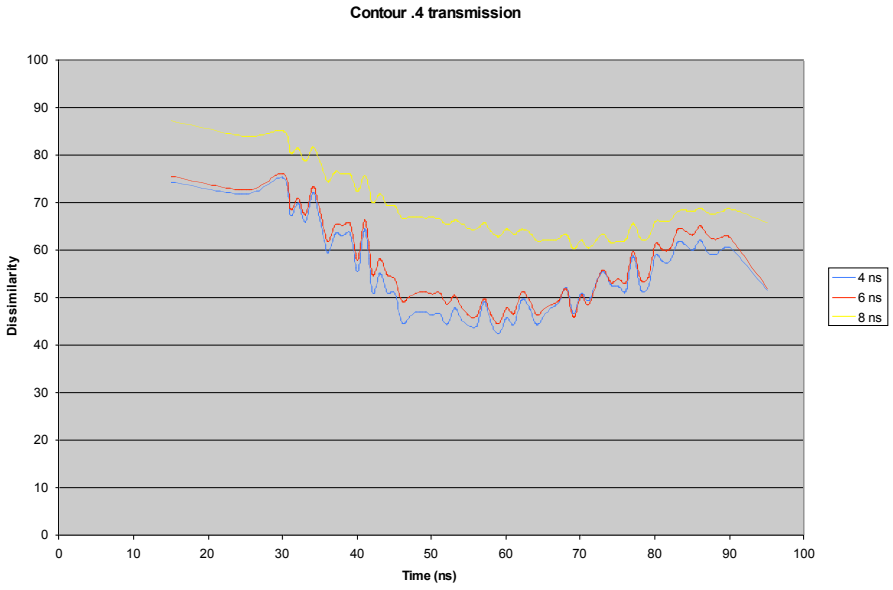

Figure 3. Results of a previous attempt to use the line scan transform to compare the jet images.

The LST approach we are studying is only one of an infinite number of ways to compare images. For example, consider N images, each being 200 pixels by 200 pixels (40,000 pixels per image). Each image can be reorganized into a vector by using the brightness values of the 200 pixels in the first row as the first 200 components of the vector, the 200 pixels in the next row as components 201 through 400 of the vector, etc., until all 40,000 pixel values are placed in the vector. Let x_i be the vector corresponding to the *i*th image. The Euclidean distance between x_i and x_j is the RMS difference between the corresponding images, and is certainly one possible distance metric, but we can weight the dimensions to come up with an infinite number of other metrics as well. Not only are there an infinite number of linear distance metrics that can be used, but image distance metrics can be (and are often) very complex nonlinear functions.

Since many different approaches can be used to compare images, it is essential that we have some valid means of comparing the results of different approaches. Typically, image comparisons are evaluated based on how well they perform on some well-defined task. For example, we might use image comparison results to cluster different images and then see whether the clusters give valuable information, i.e., do images in one cluster represent cancerous cells while the images in another represent healthy cells? Unfortunately, we do not have a clear set of criteria to meet for this task. In the discussion above, we suggested theoretical reasons that the LST approach we are developing has some advantages over other approaches for the Foster data: 1) the LST is intuitively related to mixing; 2) the LST will likely provide valuable information even when shapes are distorted by uncontrollable aberrations in the experimental set-up; and 3) we will explicitly separate information about the similarity of the different structures in the image to allow the experimenter to focus on the more relevant comparisons. However, no scientist can be fully satisfied with a result that is not subject to empirical validation. Thus, in Section 4.1.2, we make some initial steps toward understanding how to compare various distance metrics.

To compare metrics, we need to understand how a distance value relates to other distance values. Clearly, if one distance metric gives values that are always twice the value of a different metric, the metrics are essentially the same. Similarly, if one distance metric always gives values that are the square of another metric, or some other nonlinear function of another metric, the two metrics convey the same information and should be considered equivalent. Thus, an important step in comparing metrics is to first normalize the metrics to eliminate differences that are merely the result of a scaling or a non-linear mapping. We use multidimensional scaling to do such a normalization.

In addition, to determine whether a metric is consistent with our expectations, we need to be able to check the relative distances between images. For example, suppose we have three images, called I_1, I_2, and I_3, and we consider I_1 and I_2 to be very similar but I_3 is dissimilar to I_1. Now suppose that we also have two metrics, M_1 and M_2. If the M_1 distance between I_1 and I_2 is 11, and the M_2 distance between I_1 and I_2 is 2, then it might appear that M_2 is better than M_1, because it gives similar images low distance values. However, if the M_1 distance between I_1 and I_3 is 400, but the M_2 distance is .01, then M_2 would appear to be the inferior metric because the relative distances are the opposite of our expectations. Multidimensional scaling lets us visualize the distances between images, and thereby determine whether our expectations are being met.

2. The Line Scan Transform (LST)

The typical first step in comparing images is to characterize the images as a set of features. A feature can be any linear or nonlinear function of the image. For example, the feature set could be simply a list of the brightness values of every pixel in the image, or the feature set could be the positions of the transition points along an image midline, as shown in Figure 4. The distance between the feature sets for different images (calculated by whatever distance metric you choose) is then used as a measure of the dissimilarity between images.

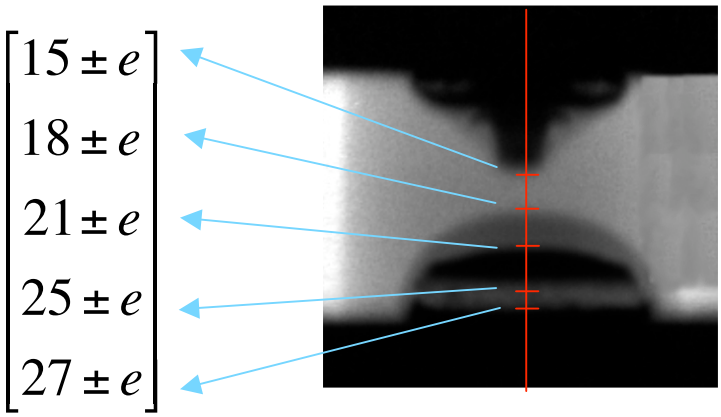

Figure 4. The transition points of the various features in an image can be placed in a vector that is then used to characterize the image.

Comparing the transition points along the midline in the simulation to the transition points along the midline in the experimental data is certainly a start at comparing the images, but it ignores the fact that the experimental image is not symmetric, and misses differences that might be observed off the midline. To ensure that the feature set is more likely to capture differences between the images, we could measure the transition points both off and on the midline. If we measured the transition points across every row or column of the image, we would have complete information on the positions of the transitions.

In some cases it is desirable to have a metric that is insensitive to rotations of the image. We can extend the idea of measuring transitions along lines to obtain a rotation invariant representation of the image. Instead of measuring transition points along only the rows and/or columns of an image, we can measure the positions of transitions along randomly placed lines. This is the basis for the Line Scan Transform, which is discussed in more depth by Cannon & Warnock (20XX).

To find the LST, we first threshold an image, setting all pixel values with a brightness greater than or equal to the threshold to white, and pixels with a value lower than the threshold to black. Then we randomly place lines across the image and measure the locations of transition points (points where we transition from black pixels to white pixels or vice versa). The transition points are used to calculate distances within black regions and distances between black regions. The LST is a curve constructed from the measured distances and which is unique to a particular image (modulo rotation). In our work, the LST is stored as a vector of 1400 values sampled along the LST curve. Thus, each image is represented as a point in a 1400-dimensional space. We use the Euclidean distance between the LSTs as a measure of the distances between images.

3. Shape-based threshold selection

Choosing a good threshold is critical for finding an LST that captures important transition points in an image. We are using a shape-based thresholding method to automatically capture an important structure in the images – the bow shock. We hope to extend this technique to allow us to capture other important structures in the future. The thresholding technique we studied is described by Kelly et al. (2006). For the convenience of the reader, the Kelly et al. paper is duplicated here with only slight modifications.

The choice of the thresholding algorithm is not only important because of the impact on the LST, but also because a major goal of automatic image comparison is capturing the knowledge of human experts and conveying it to future researchers. For example, Foster et al. compare experimental and simulation images along contours that "locate the bow shock in the cylindrical polystyrene block and times of 4 and 6 ns after the radiation drive." However, it is not clear exactly how that contour level was chosen. Undoubtedly, human expertise played a role. To some extent, the contour was determined using information about the change in transmission values in the image, but transmission values change rapidly at several points. The researchers, we speculate, also used general knowledge about the expected shape of the bow shock and the expected position of the shock to find the right contour. Our approach lends itself to capturing expert knowledge because it can be adapted to choose thresholds based on qualities of objects (such as shape or position) observed in the thresholded image.

In order to capture expert knowledge, it should be relatively easy to explain the thresholding algorithm to an expert, and then modify it to more accurately capture what the expert does. Techniques that choose a threshold based on entropy, k-means analysis, and other criteria that ignore shape do not meet this requirement — it is very difficult to incorporate human expert knowledge if the thresholds found by these algorithms are not what we want. In contrast, if a shape-based algorithm fails to work as desired on a specific set of data, it is relatively easy to incorporate additional expert knowledge in an effort to improve its performance. The shape-based approach has the additional advantage that it can be used to summarize expert knowledge and convey it to future researchers.

We restrict our attention to the most common type of thresholding function, in which each pixel in the resultant monochrome image is derived solely from the single pixel value at the same location in the original grayscale image[1]. Any gray-level pixel with a value greater-than-or-equal-to a global threshold value, *T*, is replaced with *white*, and all other pixels are replaced with *black*. Figure 5 shows an image of the moon that has been thresholded with several different values for *T*.

The ideal threshold value for a particular image is dependent on the ultimate goal that we are trying to achieve. For example, if we ask people to threshold our image of the moon (Figure 5) as a step toward determining its current phase, most people would choose a threshold value close to $T = 1$. We would be surprised if somebody chose a value $T = 192$ to accomplish this goal. In contrast, if we ask people to choose a threshold that highlights the moon's surface morphology, some people may choose a threshold value near $T = 128$, while others may select threshold values closer to $T = 192$. This discrepancy might indicate that our goal – as stated – is too vague in its attempt to define what we desire unambiguously. Further refinement of our goal can decrease the subjectivity in choosing a threshold. Continuing with the moon example, if we ask people to find a threshold that highlights surface morphology by outlining the predominant valleys of the moon, we might very well achieve consensus that a threshold near $T = 128$ is desired.

[1] More complex thresholding functions also exist, in which a single pixel in the output monochrome image is derived from several pixels in the original grayscale image. Most of these techniques are commonly referred to as adaptive thresholding algorithms.

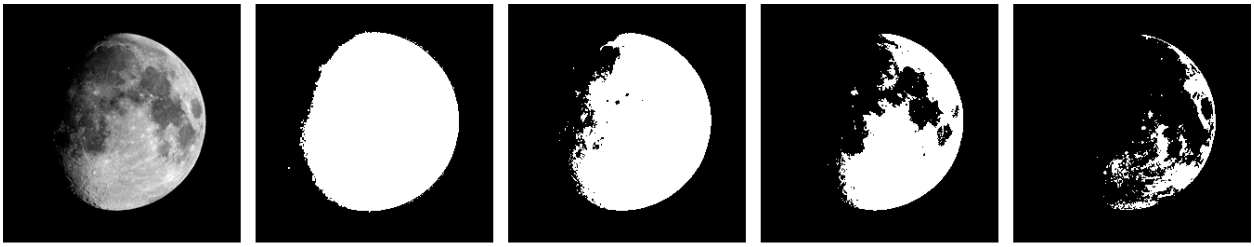

Figure 5. Thresholded images of the moon using various threshold values.

In a recent survey of image thresholding techniques, Sezgin and Sankur (2004) divide automated thresholding algorithms into six general categories. The majority of these techniques identify a global threshold value using only gray-level histogram information. Some of these methods look for peaks and valleys in the histogram, while others model the histogram as "random processes generating foreground and background pixels", and still others attempt to select a threshold that optimizes a criterion such as "information transfer". These methods rely on assumptions about foreground and background pixels and the way they appear in the gray-level histogram. The advantages to these techniques include the fact that some of them are extremely fast, and memory requirements for computation are typically minimal. These histogram methods do not, however, make use of any spatial information at all. Since human experts are often interested in the shapes of objects, and shape information is not available in a brightness histogram, it is difficult to make direct use of expert knowledge within histogram-based techniques.

Another particularly interesting category of techniques that is mentioned by Sezgin and Sankur includes methods that examine the spatial characteristics of white (or black) pixels in the resultant monochrome images. While most published algorithms in this category make use of attributes that are – in some sense – designed to be useful for any image, the general approach lends itself well to the development of specialized thresholding techniques for a given problem domain. In this sense, as long as analysts can describe what attributes they are ultimately looking for, an algorithm can be developed that specifically attempts to address their preferences. Thresholding techniques of this type are the focus of this paper.

3.1. Threshold Selection Using Connected-Component Features

Consider the example image shown in Figure 6. This image depicts four physical items (blocks) that lie within the image frame. If we threshold this image in order to perform image analysis, we would probably like to see four distinct objects with smooth boundaries clearly delineated in the resultant monochrome image. Thresholding techniques that work only through examination of the gray-level histogram, however, do not typically produce this result. Some of these techniques will isolate a single block (the brightest one), while others tend to produce an image with "partial" blocks in the image. Our goal of obtaining four distinct blocks can, however, be easily achieved interactively, where the user could select a threshold value near $T = 57$. As we look at smaller or larger threshold values, we begin to move away from this "optimal" result (see Figure 7).

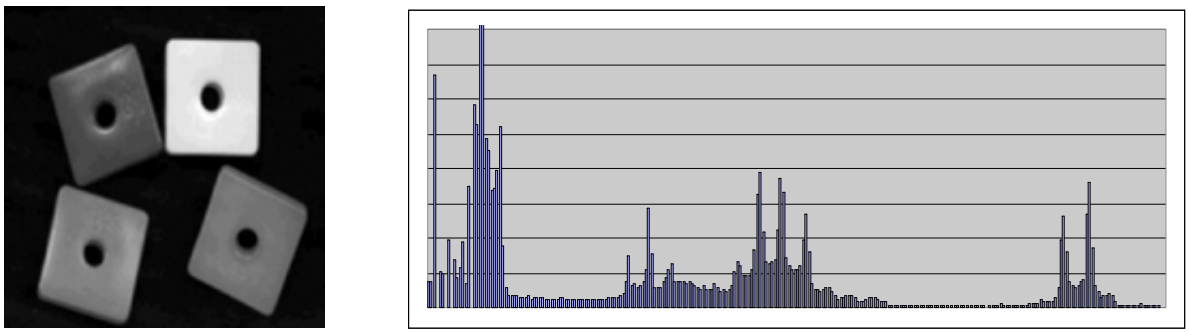

Figure 6. An example image with its gray-level histogram. Many histogram-based thresholding techniques will not produce an image where the four blocks are clearly delineated.

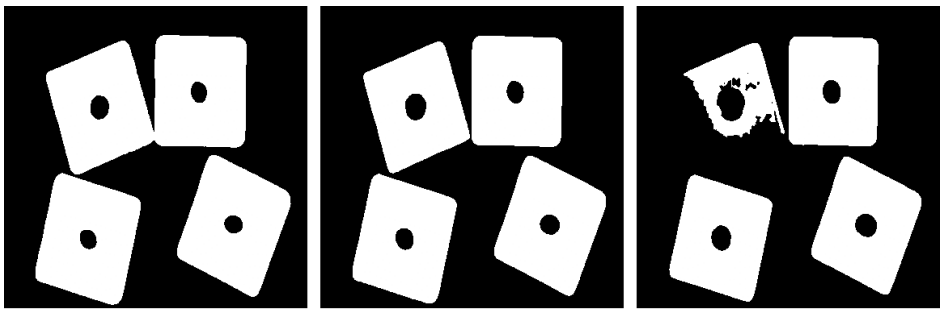

Figure 7. Interactive selection of a threshold for the "block" image. A threshold value of 57 results in four clearly identifiable blocks in the image. Smaller threshold values cause the uppermost blocks to touch, while larger thresholds cause some of the blocks to deteriorate.

In this example, we can define what we would like to achieve in a fairly straightforward manner – "*We would like to threshold this image in such a way as to see four distinct white objects*". In practice, we may find that this specification is not sufficient. A large number of possible threshold values will yield "*four distinct white objects*" in the resultant monochrome image, but most of these images do not depict the four separate "blocks" that we are ultimately interested in. Examples of this can be seen in Figure 8. A threshold value of $T = 4$ results in an image that contains three extremely small objects (each consisting of only a few pixels) and one very large object that covers nearly the entire image. A threshold value of $T = 26$ will also produce four distinct white objects, but one of these objects actually represents two blocks that are "touching" in the image, with another object simply being a small piece of "speckle". Something akin to our desired result can be obtained with $T = 55$.

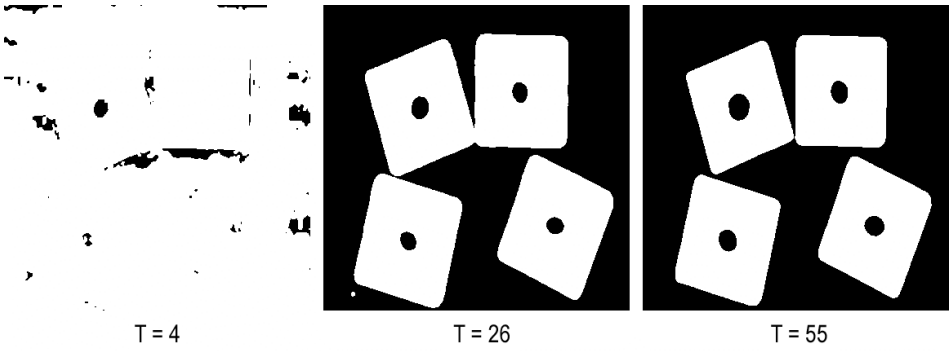

Figure 8. Designing an automated algorithm for thresholding the "block" image. Several different thresholds will produce results that depict four distinct white objects in the thresholded image, although some of these will not reflect what we were ultimately seeking.

If we can define a criterion function that is maximized when our output monochrome image depicts the four "blocks" as distinct, separate white objects in the image, then we can use this function to automatically choose a threshold value that will accomplish our desired goal. We start with the following definition for our criterion, which is a function of the input monochrome image at threshold *T*:

$$f(\mathrm{I_T}) = \begin{cases} \sum_{j=1}^{4} \#\mathrm{pixels}_j & \text{if } \#\mathrm{objects} = 4 \\ 0 & \textit{otherwise} \end{cases}$$

The plot of this function for our "block" image example is shown in Figure 9. Every threshold that produces a monochrome image with exactly four distinct white objects (e.g. four connected components) will produce a non-zero value of this criterion function. The values on this plot are simply the total number of pixels contained in the four white objects. If we choose the threshold value *T* that maximizes this function, our result will be *T = 4*, which is obviously not the answer we desire.

Our remedy for this situation can be a simple bit of preprocessing built in to our criterion. Specifically, if our function ignores very large objects (such as those that touch one of the image borders) as well as very small objects (such as those containing fewer that 100 pixels), then it will successfully select *T = 55* as our desired threshold (see Figure 10).

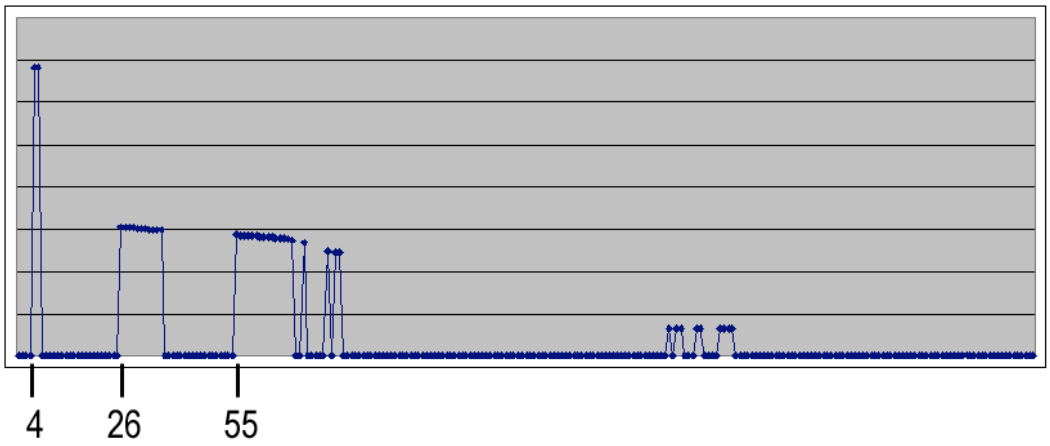

Figure 9. Criterion function values as our threshold varies from 0 to 255. Valid points (non-zero) on this plot exist for thresholds where four distinct white objects are visible in the "block" image. Thresholds that yield some other number of objects produce a criterion value of zero.

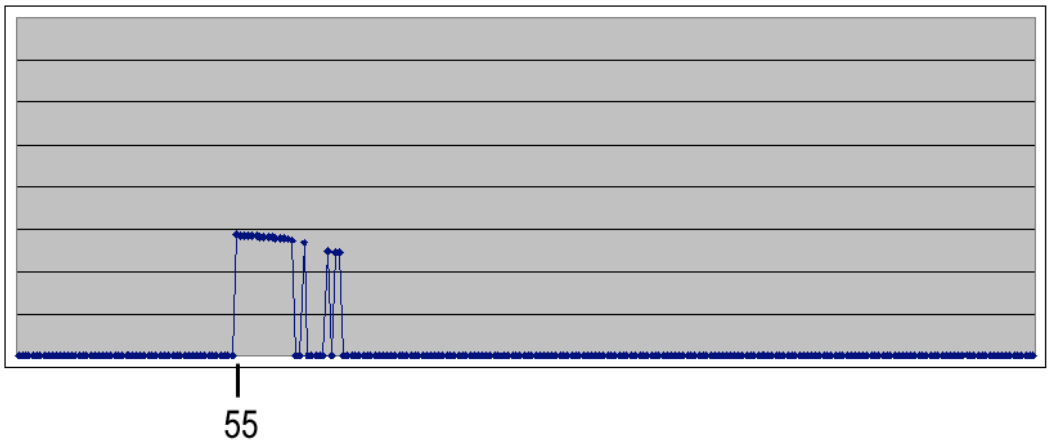

Figure 10. Criterion function values when very small/large objects are ignored in the image.

This same methodology can be used to incorporate expert knowledge into a thresholding routine for almost any application. Generally speaking, the criterion function should be specialized enough to "get the job done", while not being overly restrictive (which might cause it to be "too specific" and, as a result, fail when presented with new data).

3.2. Automated Thresholding of Hydrocode Simulation Data

As with the moon example above, when we consider the problem of analyzing hydrocode simulation data, different threshold values may result in images that capture distinctly different aspects of the underlying data. An example of this can be seen in Figure 11, where different threshold values have been used to generate monochrome imagery depicting the "mushroom jet" and the "bow shock" for a specific frame in a simulation.

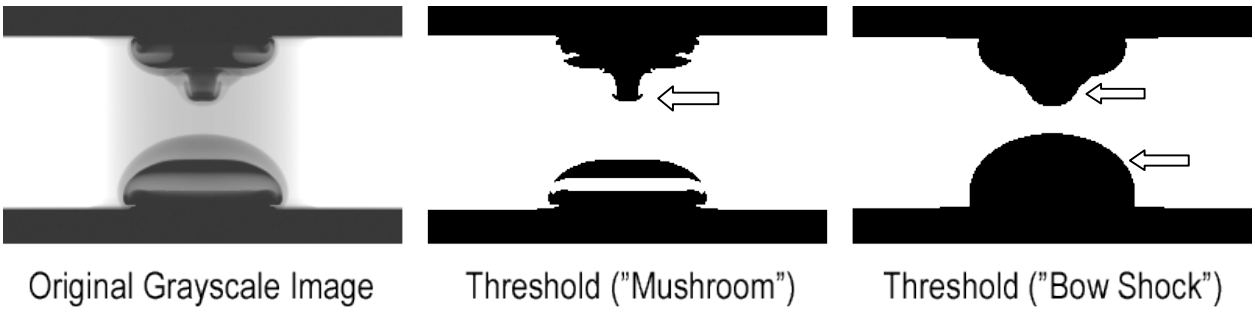

Figure 11. Different thresholds are needed to extract the jet and the bow shock from the image

It has been suggested that the relative positions of the jet and the bow shock are important for characterizing these images (Wilde, 2005), so our goal is to find two or more thresholds for each image — one threshold that can be used to extract the bow shock and another to extract the jet. For purposes of this paper, our attention is restricted to finding threshold values that result in monochrome images depicting bow shocks in the simulation.

Using the principles described in the preceding section of this paper, we tried to address to the specific problem of thresholding hydrocode simulation data by asking ourselves a simple question – "What do we look for when we threshold this image by hand?" Our answer was then used to build an automated algorithm that would mimic our interactive approach to choosing an appropriate threshold value. It was important in this exercise to operationally define what we were looking at in terms of the image. Even if our automated algorithm ultimately fails in its attempt to characterize what human experts would do, it still provides a starting point that can be refined.

Our observation was that – when thresholding these images by hand – the images exhibited specific properties when we were near the desired threshold. As we moved our threshold from a large value near 255 (where the image is nearly all-black) to a small value near 0 (where the image is nearly all-white), there was always some point at which one of two things happened. In one case, the white region in the middle of the image would suddenly connect all the way from the left-hand edge to the right-hand edge. This phenomenon would occur in images where the bow shocks were still distinctly separated. In the second case, black areas would begin to "fracture" in some way (typically with the appearance of small white holes where the density inside of the bow shock was not distinguishable from the density outside of the bow shock). This would occur in images where the bow shocks had already come together (see Figure 13). Using this knowledge, we were able to build an automated threshold routine that can successfully find bow shocks in this set of simulated data. It is an open question how well these techniques will generalize to future data, but it is a question we hope to explore further.

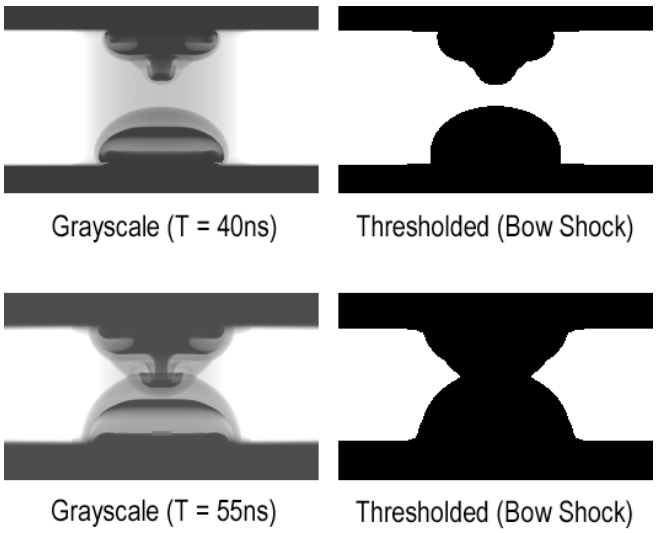

Figure 12. Selecting threshold values for hydrocode simulation data in an effort to generate monochrome images that depict "bow shocks" for analysis.

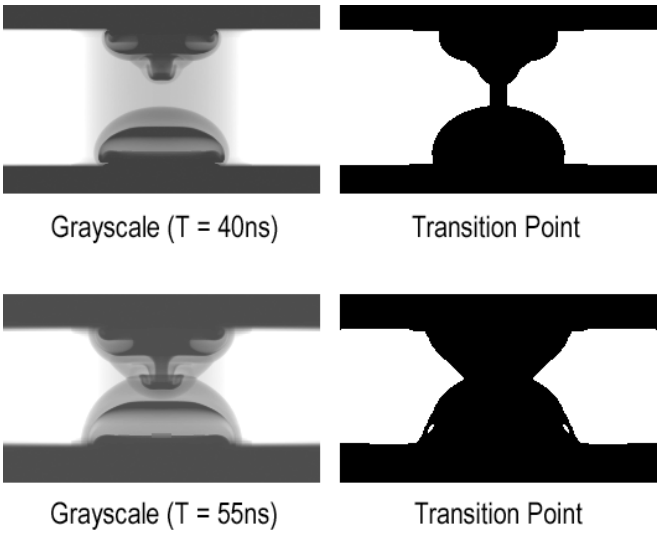

Figure 13. Threshold values just past the "transition point" for the case where (A) the simulated bow shocks had not yet come together; and (B) where the simulated bow shocks had already merged

4. Comparing experiments with simulations

Figure 14 shows the results of comparing the experimental images to each of the simulation images. The x-axis indicates the simulation time, so the leftmost plot tells us that the 4 ns experimental image is most similar to the 4 ns simulation image, but is also very close to images at 4.1 ms and 4.2 ns. Note that the y-axis does not go to zero in any of the plots, indicating that, as we expect, the experimental images are not identical to any of the simulations.

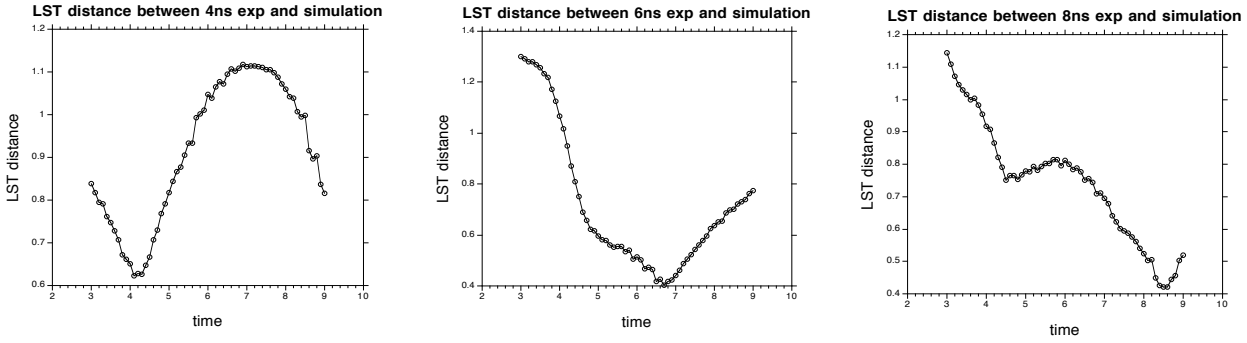

Figure 14. These plots show the distances between the experimental images and each of the simulation images.

The first thing to notice about Figure 14 is the dramatic change from the results of the earlier application of the LST shown in Figure 3. All of the curves in Figure 14 are much smoother than the comparable curve in Figure 3, the global minima are more distinct, and the timing of the global minima is much closer to what we expect. The differences between Figure 3 and Figure 14 are attributable to the differences in threshold selection techniques that were employed. Recall that, in the earlier results, the threshold was set to a constant 0.4 for all images. Since the transmission values for structures vary over time, the 0.4 transmission value does not adequately track structures. The result is that different structures are being compared over time if the threshold is constant over time. In contrast, our current technique finds different thresholds at different times in an effort to track a structure (the bow shocks). Clearly, tracking is an important step. We hope to improve our tracking techniques in future work by adding more expert knowledge, and by developing criteria to track more structures in the images.

While the 4 ns experimental image is most similar to the 4 ns simulation, the 6 ns experiment appears to be most similar to the 6.7 ns image, indicating that the simulation lags behind the experiment in some feature. Similarly, the 8 ns experiment seems to most closely match the 8.6 ns simulation.

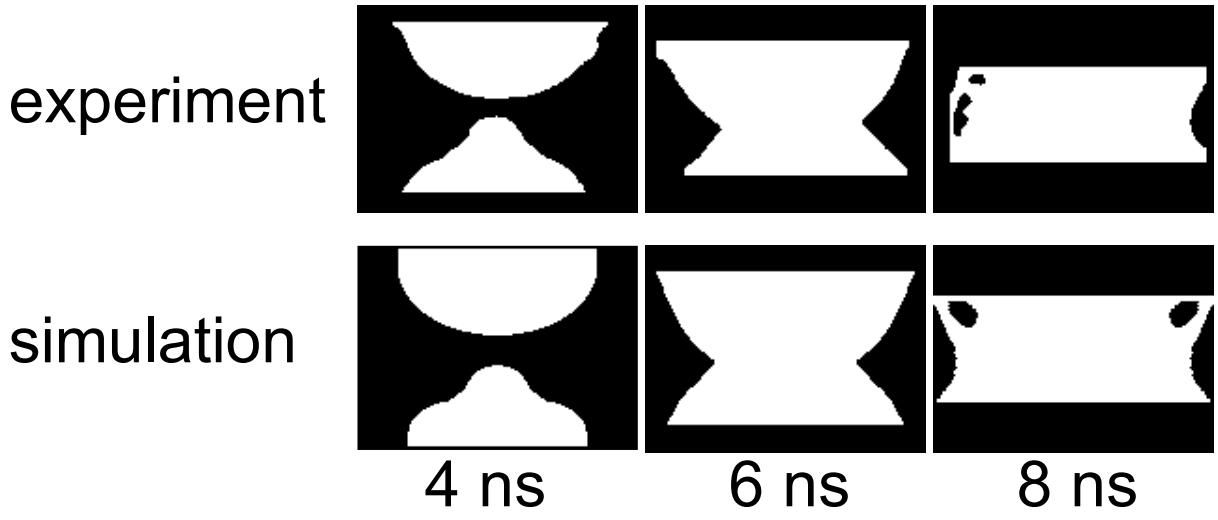

Figure 16. Thresholded experimental images used for comparison to simulations.

Note that the 4ns experimental images become more similar to the simulations as the simulation time increases beyond about 7 ns. Furthermore, the 8 ns plot shows a distinct local minima around 4.5 ns. These two facts tell us that the similarity measure is finding some feature in common between the 4 ns images and the late-time images. Figure 16 shows the thresholded images for the experimental and simulated data at 4, 6, and 8 ns. In the 4 ns experimental and simulation images there are two white regions, each of which has a height that is nearly the same as the single white region in the 8 ns images. In contrast, the 6 and 8 ns image has a single white region that is much taller than the regions in either the 4 or 8 ns images. The region height is undoubtedly a feature that makes the 4 and 8 ns images more similar than either is to the 6 ns image, and the number of white regions (one versus two) is a feature that distinguishes the 4 ns image from the 8 ns image.

The fact that the 6 ns experimental image most closely matches the 6.7 ns simulation image instead of the 6ns simulation image is a bit surprising. It seems likely that the automated thresholding did not perfectly duplicate expert knowledge, which may account for some of the differences. However, other differences between the simulation and the experimental image are also apparent in Figure 16. For example, the 6 ns experimental image is not symmetric and the black regions below and above the white region are different sizes in the experiment than in the simulation (the difference can be explained by differences in the position of the gold from the washer).

To help understand the impact of thresholding and the asymmetry, we created variations on the experimental images. We split each experimental image along the axis of symmetry so that, in a perfectly symmetric image, the two halves would be reflections of each other. Since the experimental images are not perfectly symmetric, the halves are not

actually reflections. Each of the half-images was then converted into a perfectly symmetrical image by reflection. Thus, we converted the three original experimental images into 9 images — three original images, three right sides, and three left sides. These images were all thresholded at three different thresholds — our best guess threshold, one brightness level higher, and one brightness level lower. The result was 27 quasi-experimental thresholded images. The quasi-experimental images at the best guess threshold, along with the corresponding simulation images, are shown in Figure 17.

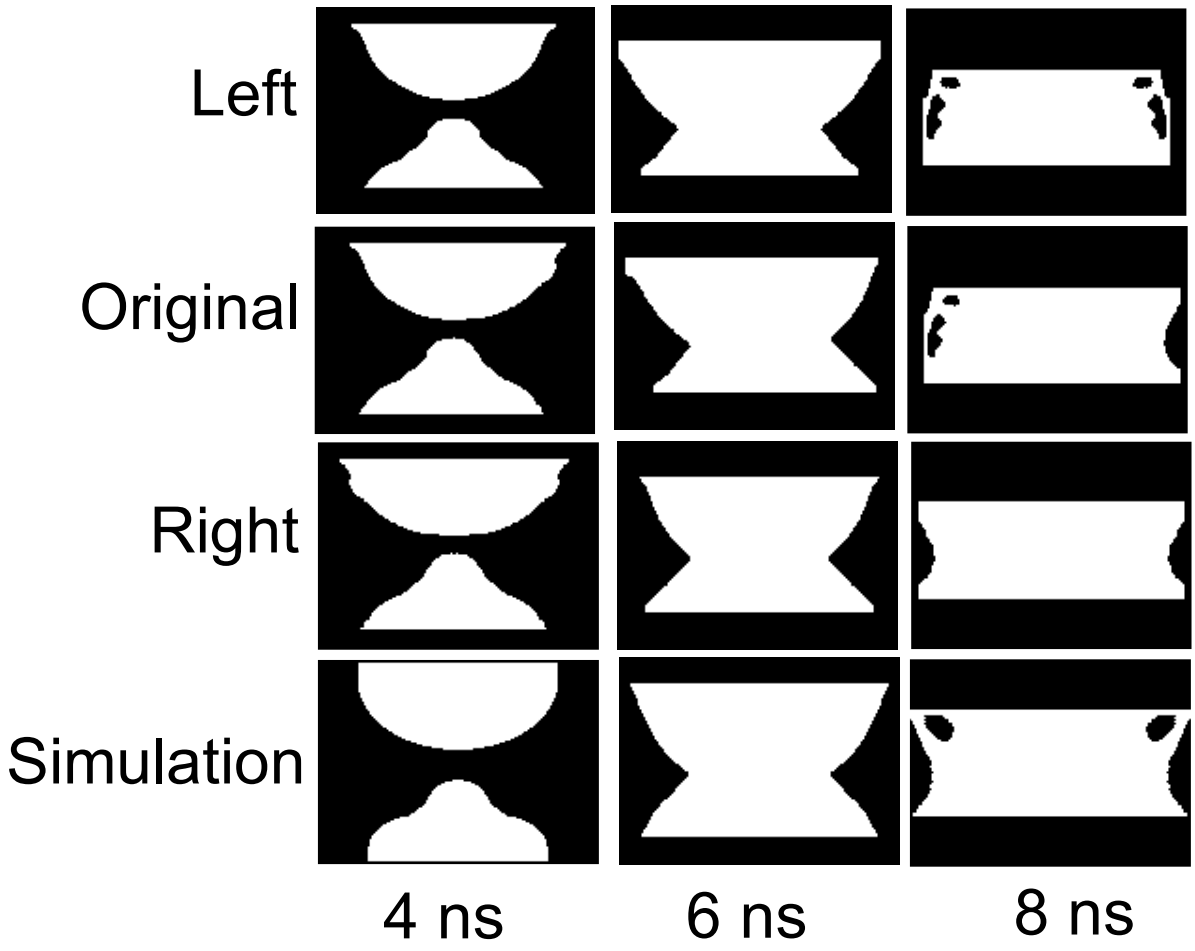

Figure 17. The left and right halves of the asymmetric thresholded original experimental images are transformed into new symmetric images and shown along with the symmetric simulation images.

Comparing each of the 27 quasi-experimental images to the 61 simulation images results in 1647 distance measurements. However, we would also like to have information about how the distance between quasi-experimental images relates to the distances between the quasi-experimental images and the simulations, and the distances between each simulation and each other simulation. To get all this information, we compared every image to every other image for a total of 3,828 distance measurements.

4.1. Multidimensional Scaling

Making sense of 3828 distances requires visualization techniques. In this section we give a brief discussion of multidimensional scaling (MDS). In section 4.1.1 we discuss MDS as a technique for visualizing distances between points. In section 4.1.2 we go beyond visualization to discuss how MDS may be useful for comparing different similarity measures. Finally,in Section 4.1.3 we give the results we obtained when we applied MDS to the image distances.

MDS takes as input a set of measured distances between points (in this case, the LST distances between images) and outputs a solution that gives an estimated position for each point in a space of some specified dimensionality. In *metric* versions of MDS, the positions of the points are found so as to minimize the root mean squared difference between measured interpoint distances and the interpoint distances estimated by MDS. Metric versions of MDS will typically give different solutions if the distances input to the algorithm are monotonically transformed, e.g. if the square of the distance is given instead of the distance. In contrast, *nonmetric* versions of MDS attempt to position points in a multidimensional space so that the interpoint distances in the MDS solution are a monotonic function of the input distances. The fit between the input interpoint distances and the interpoint distances in the MDS solutions is measured by an error term called stress, which typically decreases as the dimensionality increases. Since nonmetric MDS tries to find the best fit to within a monotonic function, it tends to give very similar solutions even if the input distances have been transformed by a monotonic function.

4.1.1. Multidimensional scaling for visualization

As described in Section 2, the LST transforms each image into a 1400-dimensional vector. We then calculate the distances between the images as the Euclidean distance between the corresponding vectors. However, the number of distances between points is typically much large than the number of points. To be precise, if we have N points then we can calculate N(N-1)/2 distances. While the ultimate goal is to understand the relationships between distance measurements, the positions of the points in the 1400-dimensional LST space gives us all the information about the distances. Therefore, being able to view the positions of points (a difficult task in a 1400-dimensional space) can help us understand the distance information more easily than looking at a large amount of distance information.

A priori, there is good reason to believe that the positions of our images in the LST space can be viewed using much fewer than 1400 dimensions. Consider that differences between simulation images can all be attributed to differences in time — no other variables were changed in the simulation data we analyzed. Any particular image in the simulation is, therefore, some nonlinear function of time. This implies that, even though the simulation images are represented as points in a 1400-dimensional space, they actually lie on a one-dimensional curve. The situation is completely analogous to what happens in mappings in lower-dimensional spaces, such as a mapping from R^1 to R^2. An example of such a mapping is shown in Figure 18. In this example, the one-dimensional domain of the function, time or t, varies from 0 to 10. Each time is mapped onto the two-

dimensional range specified by [y_1(t), y_2(t)], using the equations $y_1(t) = \log(x(t)+1)$ and $y_2(t) = x^2(t)$. Note that the mapping does not increase the intrinsic dimensionality of the domain—the domain is mapped to a one-dimensional curve embedded in the two-dimensional space. Similarly, since the mapping from time to LST is a mapping from R^1 to R^{1400}, the domain can be represented by a one-dimensional curve embedded in the 1400-dimensional LST space. Call the curve the constraint surface.

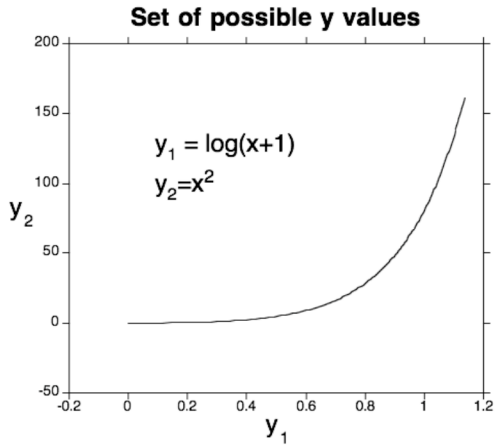

Figure 18. This shows an example of a mapping from R^1 to a curve in R^2.

In general, the constraint surface can be very complex. It could curve back on itself or form loops in many dimensions making it impossible to find a low-dimensional representation of the surface that accurately captures the distances between points, or even the general ordering of the simulations, i.e., that the simulation at time 30 is close to the simulation at time 31. Furthermore, although the simulation images can be thought of as a function of 1 variable, the experimental data cannot. Asymmetry in the experimental images (and common knowledge) attest to the fact that some aspects of the experiment were beyond control, and so can be thought of as free variables. Differences in the thresholds chosen for the experimental images can also be thought of as the result of a free variable. Thus, we are not guaranteed to be able to find a low-dimensional representation of the data. While we cannot prove this point, we speculate that smooth similarity functions, such as those we obtained and show in Figure 14, will typically allow low-dimensional solutions.

There are a variety of methods for finding low-dimensional representations of points in high-dimensional spaces. Researchers commonly use principal components analysis (PCA) for this purpose. Local Linear Embedding and ISOMAP are becoming more popular for this purpose as well. However, none of the techniques mentioned above are optimal for our purposes. PCA is limited to only allowing linear projections of the points. LLE and ISOMAP allow nonlinear projections, but only retain distance information between points that are close to each other, whereas we also want to understand how the LST places points that are farther from each other.

Multidimensional scaling is yet another technique that can be used to find a low-dimensional representation of points. MDS is particularly applicable to this task because

it attempts to accurately capture the distances between the images. Furthermore, since the fit between the distance measurements and the positions of points in the MDS solution is measured along a monotonic function, MDS often captures distance information using fewer dimensions than PCA.

4.1.2. Multidimensional scaling for normalization

Applying a monotonic function to distances can have such a large effect on image similarity measures. For example, the curves shown in Figure 14, which give the distances between experimental images and simulations, are obtained using a Euclidean distance metric between points. By simply applying a monotonic function to the Euclidean distance, we could make a large class of similarity measures that would appear to be different, but which are all based on the same image features. For example, Figure 15 shows how Figure 14 could be changed by applying a monotonic function to the distance values.

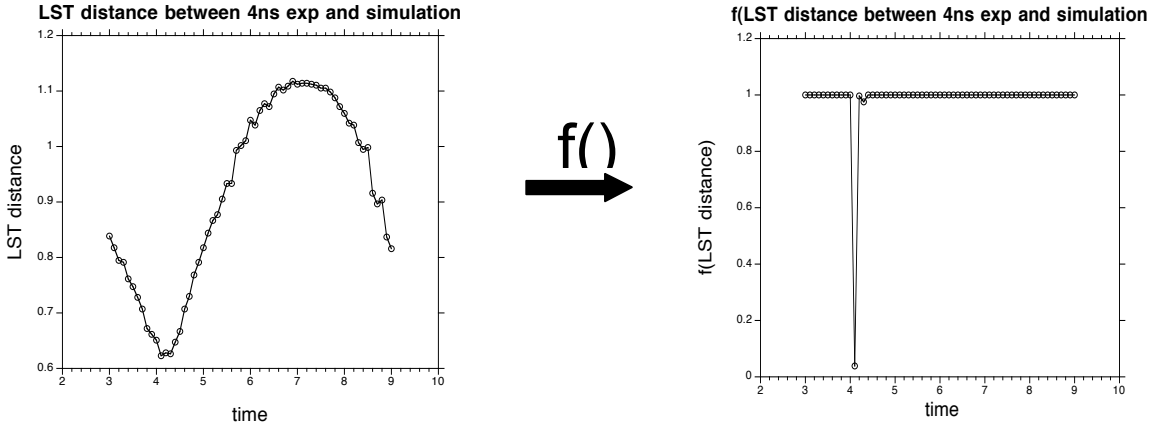

Figure 15. Applying a monotonic function to distances between points can give a similarity measure that appears to be very different despite being based on the same image features

If we were to compare the similarity measurements shown in these two graphs, we might choose one over the other on the basis of the shape of the graph, e.g., the similarity measure shown on the left might be preferred because of the more gradual increase of dissimilarity as we move away from the lowest dissimilarity value, or possibly we would prefer the similarity measure with the graph shown on the right because the local minimum problem appears to be diminished. However, the similarity measurements used to produce the left graph actually have the same information content as the measurements used to produce the graph on the right, since they are merely transforms of each other.

The fact that applying a monotonic function to distances can have such a large effect on image similarity complicates the problem of comparing metrics. We must think not only about what qualities are desirable similarity measurement, but must also consider which

other similarity measurements can be made to have those desirable qualities by a simple application of a monotonic function to the distance metric.

Nonmetric MDS may alleviate the problem of comparing similarity measurements by allowing us to normalize out monotonic functions applied to similarity measurements. Since the distances between points in the MDS space are largely invariant to monotonic transformations of the similarity measurements, comparisons of MDS distances should give a clearer idea of the relationships between similarity measures. This is a topic we hope to pursue in future work.

4.1.3. Multidimensional scaling results

Figure 19 shows the fit between the inter-image distance measurements made using our method and the distances between points in the MDS solution for both the 2 and 3-dimensional solutions. The solid curves in both plots show the monotonic function that gives the best fit. In general, to determine how many dimensions are needed, we calculate the stress for solutions with differing numbers of dimensions and choose a dimensionality that captures the distances accurately enough. In this case, it can be seen that the three-dimensional solution is capturing the distances very accurately, although with somewhat greater error for LST distances between about 0.3 and 0.5.

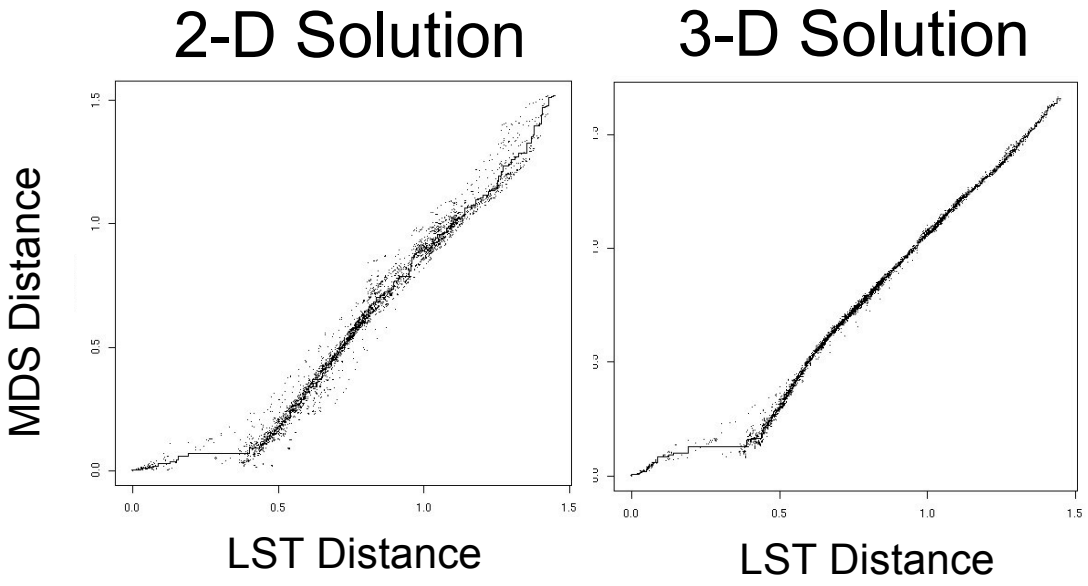

Figure 19. These functions relate the Line Scan Transform distances between images to the distances in the two- and three-dimensional multidimensional scaling solutions.

We also created MDS solutions in higher dimensions. The stress values obtained for solutions of various dimensionality are shown in Figure 20. Note that the stress does decrease with increasing dimensionality but that there is relatively little reduction in stress in moving from the three-dimensional solution to a four-dimensional solution. This suggests that the distances between points in the three-dimensional solution is a good approximation of the distances between the points in the 1400-dimesional LST space.

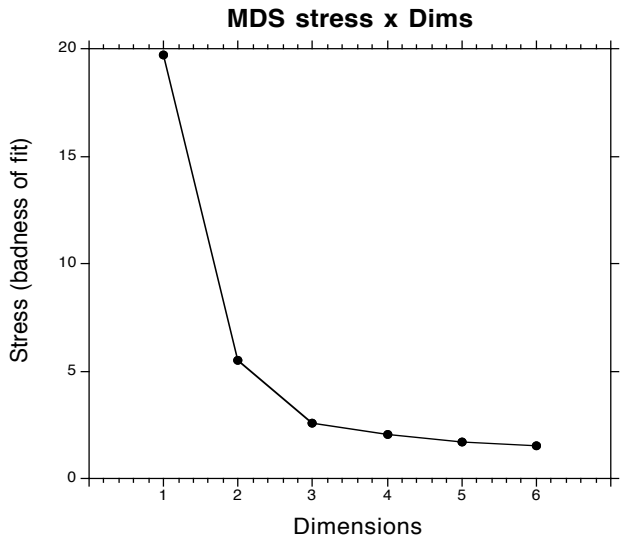

Figure 20. The stress of the MDS solution decreases with as the dimensionality of the solution increases.

Although the two-dimensional MDS solution less accurately captures the distance information than the three-dimensional solution, a comparison of the two- and three-dimensional solutions not only clarifies the structure of the three-dimensional solution, it also shows how the inaccuracies of solution can be misleading. Figure 21 shows the two-dimensional solution and Figure 22 shows the three-dimensional solution. The numbers preceded by an "S" in Figure 21 give the time, in tenths of nanoseconds, of a simulation image. For example, "S31" is the position found for the simulation at 3.1 ns. The one-dimensional curve that captures the change in time can clearly be seen in both the two- and three-dimensional solutions — "S30" is on one end of the curve and "S90" is on the other with the other simulations shown in roughly the correct order along the curve.

In the two-dimensional solution, the experimental images (preceded by an "E" in figure 21) are tightly clustered off the simulation curve for both the 4 ns and 6 ns images, but the 8 ns images overlap the simulation curve. The tight clustering of the 4 ns experimental images is not seen in the three-dimensional solution. The three-dimensional

solution shows the 4 ns and 6 ns images roughly arranged along a line perpendicular to the curve containing the simulation images.

The tight clustering of the 4 ns and 6 ns images is an artifact of projecting a high-dimensional space down to two dimensions. For example, in Figure 22A we see that the 4 ns positions appear to be tightly clustered when viewed at one angle, even though the points are clearly not clustered when seen from the other angles shown in the figure. That the 4 ns images are not, in fact, tightly clustered can also be seen by comparing the distances between the various modifications of the experimental images at 4 ns (right side, left side, different thresholds, etc) to the distances between the simulations and the experiments. For example, we know from Figure 14 that the 4 ns experimental image is most similar to the 4 ns simulation image. The distances between the 4 ns simulation image and the various versions of the experimental image ranges from 0.63 to 0.73. Distances between the experimental images range from 0.03 (the distance between the original experimental images using the best guess threshold and the original image using a threshold that is on level higher) to 0.41 (the distance between the left image using the best guess threshold and the right image using the high threshold). So some of the experimental images have a distance of 0.41, which is more than half as large as the distance between simulation and experiment, but those distances are not accurately reflected in the 2-D solution. For the 4 experimental images, the three dimensional solution more accurately captures the main differences between the images, which is the difference between the right and left sides.

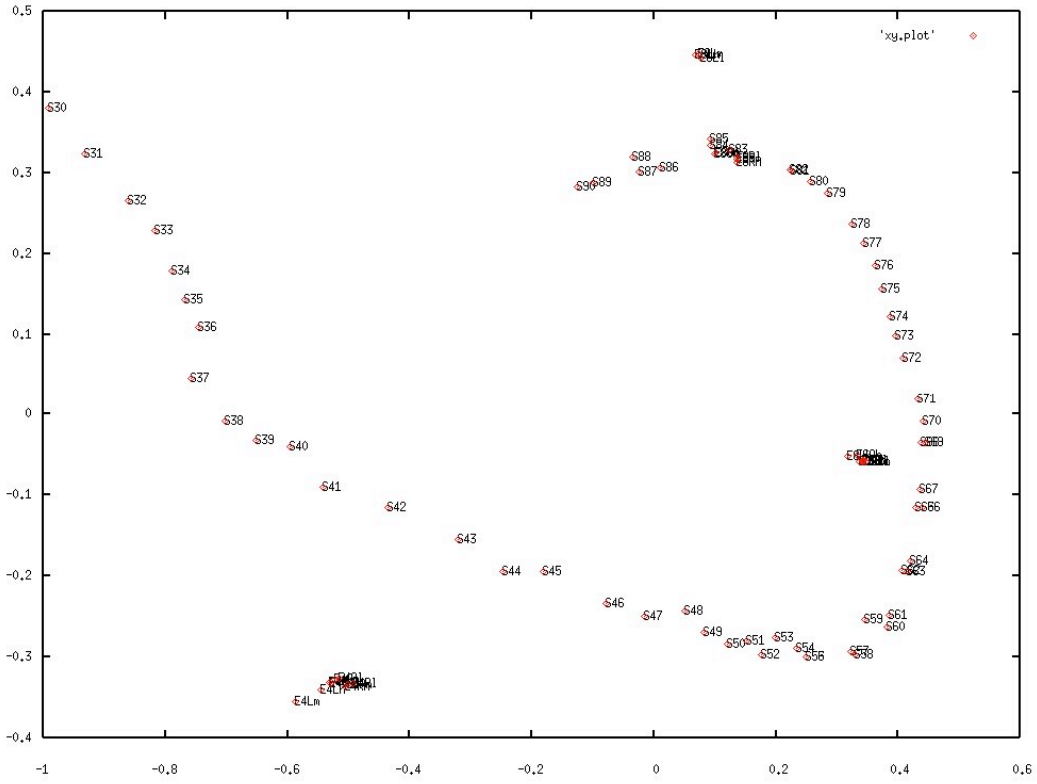

Figure 21. the two-dimensional MDS solution. The numbers representing positions of simulation images are preceded by an "S". Experimental images are preceded by an "E".

The story for the 6 ns experimental images is different from that of the 4 ns experiment, but, again, the apparent tight clustering of the points in the two-dimensional solution is misleading. The distribution of the 6 ns points is much better captured by the three-dimensional solution. From Figure 14 we see that the 6 ns experimental image is most similar to the 6.5 ns simulation image. The distances between the 6.5 ns simulation image and the various images derived from the 6 ns experimental image range from 0.4 to 0.46. Distances between the various 6 ns experimental images range from 0.02 to 0.15. So, the 6 ns experimental images are clustered more tightly than the 4 ns images, but the two-dimensional MDS solution still shows the experimental images to be clustered more tightly (compared to the distance between the experiment and the simulation) than they should be. The three-dimensional MDS solution shows the 6 ns experimental images spreading out along a line that is nearly perpendicular to the curve of the simulation images.

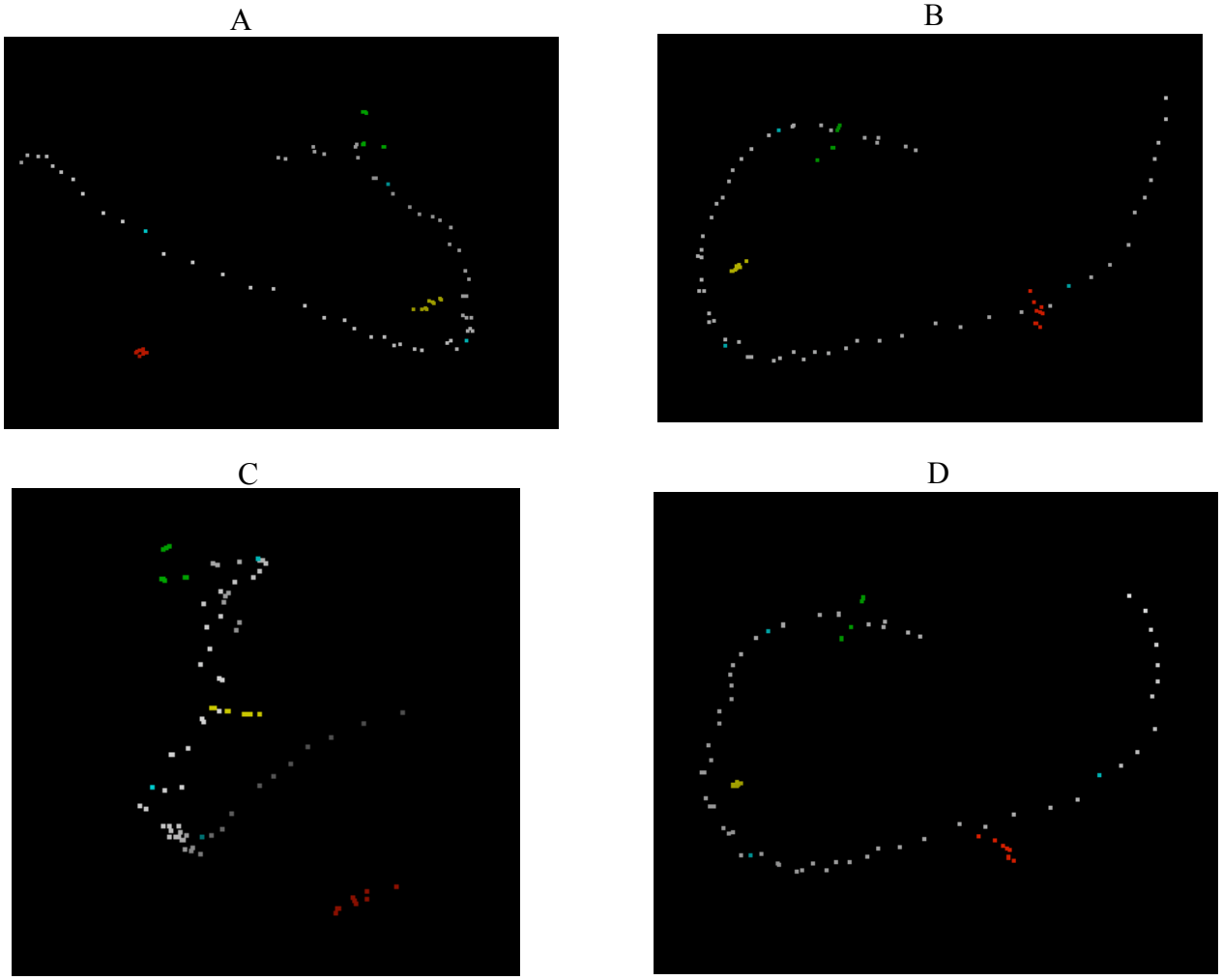

Figure 22. Different views of the 3-Dimensional MDS solution. The 4 ns experimental images (right half, left half, original, with threshold variations) are shown in red. The 6ns experimental images are shown in yellow, and the 8ns experimental images are shown in green. Simulation images are shown in shades of gray, except that the 4ns, 6ns, and 8ns simulation images are shown in blue.

A notable difference between the MDS solutions (both 2-D and 3-D) is that the 4 ns experimental images appear to be more time-delayed in the MDS solutions than shown in Figure 14. Consider that Figure 14 shows that the 4 ns experimental images is closest to the 4 ns simulation, but the 2-D MDS solution shows the 4 ns experimental images to be closest to the 4.2 or 4.3 ns simulation. At least part of the inaccuracy of the MDS solution is attributable to the fact that the 4 ns experimental images become increasingly similar to simulation images after 7 ns — the MDS solution attempts to place the 4 ns experimental images close to both the 4 ns simulations and the later simulations, with the result of pulling the later-time simulations closer to the 4 ns simulations and pulling the 4 experimental images closer to the later time simulations. This shift isn't as apparent in the 2-D solution in which the 4 ns experimental simulation images lie between the 4 ns experimental images and the 8 ns simulations, but is more readily seen in the 3-D solution.

Figure 21 shows that the 6 ns experimental image is closest to the 6.5 ns simulation, but the 2-D MDS solution shows the experimental images to be closer to the 6.8 or 6.9 ns simulation. The problem largely disappears in the 3-D solution, in which the 6 ns experimental images lie on a line roughly perpendicular to the simulation curve and appear to approach the simulation curve at about 6.6 ns.

5. Conclusions

It is fairly clear from the positions of the simulation images in the multidimensional scaling solutions that the temporal evolution of the shock waves was captured by the similarity measure. Furthermore, since the positions of the 4 ns and 6 ns experimental images were perpendicular to but did not intersect the curve formed by the simulation images, it is also clear that asymmetry and thresholding errors played a large role in differentiating the experimental images from the simulation images, but are not sufficient to explain all the differences. In fact, differences between images due to the choice of threshold and due to image asymmetries are almost as large as the differences between experimental images and simulations. It is notable that changing thresholds produced changes in the MDS solution that were in the same direction as the changes due to asymmetries. This agrees with the visual impression that the two sides of the experimental images differed in the extent of the propagation of the bow shock. Some of the differences between the experimental and simulation images were due to inaccurate modeling of the gold washer. This difference is not likely to be of significance, so in future work it would be an advantage to devise a metric that successfully separates the effect of the gold washer from other differences between images.

Choosing a threshold that varied over time so as to track features important to experimenters was apparently a big advantage. The curves comparing experimental and simulation images obtained using the present metric showed the temporal progression much more clearly (as evidenced by the increased smoothness of the similarity curves and the locations of the minima). Making stronger claims about how well the present similarity metric did compared to past techniques is difficult without normalizing the dissimilarities using MDS (or some equivalent) analysis on both present and past results.

Bilbiography

Cannon, T. M., Warnock, T, (2004) A shape descriptor based on the line scan transform. Los Alamos National Laboratory Technical Report, LA-UR-04-5865

Cannon, M., Hogden, J., Warnock, T., Fasel, P., and Fortson, R., "Statistical Methods for Analysis and Anomaly Detection in Large Hydrocode Data Sets," Nuclear Explosives Computation and Design Conference (NECDC 2004), Livermore, CA, October 4-8, 2004, LA-UR-04-3930.

Cannon, M., Coker, R., Fisher, K., Fortson, R., Hogden, J., Warnock, T., Wilde, B. Kamm, J., Perry, T., Rosen, P., (2005) Image Quantification of Experiments and Simulations of Laser-Driven Supersonic Jets, LA-UR-05-1441.

Foster, J. M., Wilde, B. H., Rosen, P. A., Perry, T. S., Fell, M., Edwards, M. J., Lasinski, B. F., Turner, R. E., and Gittings, M. L. (2002) Supersonic jet and shock interactions, Physics of Plasmas, 9(5), 2251-2263, May 2002.

Patrick Kelly, Pat Fasel, John Hogden, James Howse, and Richard Fortson (2006). Towards automated threshold selection for hydrocode simulation imagery. LA-UR-06-3899.

Sezgin, M., and Sankur, B. (2004) Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging* 13(1), 146-165, January 2004.

Wilde, B. H. (2005). Personal communication.