Automatic Deformation Detection for Aircraft Engine Disk Inspection
Dirk Padfield, Glen Brooksby, and Robert Kaucic
GE Global Research, One Research Circle, Niskayuna, NY, 12309
Email: {padfield, brooksby, kaucic}@research.ge.com

Abstract—Computer vision algorithms are seeing increased use in industrial inspection applications. Here, we present an “Aid to Visual” system that can detect post deformations of less than 0.005 inches in jet engine high pressure turbine disks. We create a gold-standard reference post from the posts of sample turbine disks and then use registration, edge detection, and curve-similarity algorithms to identify unacceptable post deformations. We address the challenges associated with adapting academic algorithms for use in functioning inspection systems. We present novel solutions to deal with practical issues such as accuracy, speed, robustness, and ease of use. We also present a novel, highly-efficient sub-pixel contour matching algorithm and demonstrate the effectiveness of using sub-pixel distance calculation. We demonstrate overall error rates less than 1% on over 2400 images of posts. We have integrated our algorithms into the commercial LabVIEW software running on the Aid To Visual workstation. Our algorithms will enable plant-factory inspectors to identify minute post deformations that were previously difficult to detect.

Index Terms—Industrial inspection, aircraft engines, deformation detection, registration, Chamfer distance, curve matching, sub-pixel distance calculation.

I. INTRODUCTION

The increase in computational power and the proliferation of cheap cameras has led to a rapid growth in the use of computer vision techniques in industrial inspection applications. For example, Gunatilake et al. [1] automatically detect surface cracks on aircraft wings in visual imagery using multi-scale edge detection algorithms. Novak and Hocenski [2] use local binary pattern (LBP) texture operators to identify defects in ceramic tiles. In [3], Acciani et al. revisit surface-mount PCB inspection by using wavelets and geometric features with a multi-layer perceptron neural network for classification to improve the visual inspection.

Here we present an “Aid to Visual” (ATV) system for inspecting jet engine high pressure turbine (HPT) disks. HPT disks consist of a ring of dovetail slots separated by posts that hold turbine blades in an aircraft engine. The disk must be made to high precision to ensure that the blades fit snugly into them. The inspection of these posts has traditionally been carried out manually, using a hand light, calipers, and a magnifying glass. To assist the inspector in this task, an ATV inspection system was developed that automatically rotates the disk, takes pictures of the posts and the dovetail slots, displays the posts on a computer screen, and makes several measurements of the posts. A prototype Aid To Visual workstation is shown in Figure 1, where an image of a post can be seen on the display screen in Figure 1(a), and Figure 1(b) shows a close-up of the system.

Figure 1. Prototype ATV workstation and camera setup. The 80 posts are on the outer edge of the disk on the middle table. The cameras are mounted above the disk and photograph each post as the disk is rotated. The user inspects the post on the computer screen for deformations and defects.

Because of mechanical drift in the positioning subsystem, the post’s location in the field of view varies from post to post. This makes it difficult for human inspectors to compare one post with another and to find small deformations in the posts. In addition, the broaching of the posts is a well-honed process, and hence the inspectors pass nearly every post. However, small deformities in the posts can occur from mis-handling or other errors in the manufacturing process. These minute deformations are

Corresponding author: Dirk Padfield

© 2007 ACADEMY PUBLISHER
very difficult to detect from visual inspection.

As an illustration, Figure 2 shows some typical images of posts. The post on the lower left is deformed and out of tolerance; it has been intentionally over-broached by 5 mils (0.005 inches). However, it is extremely difficult to see this because there is so much variability among the locations and appearances of the posts. The over-broaching results in a slightly narrower post and hence a thinner edge on the bottom. This post is shown again in Figure 3(b) and 3(d) along with a normal post with similar appearance in 3(a) and 3(c). The figures show how slight the deformation is; it is difficult to see the deformation, even in the zoomed-in version 3(d) with a template contour overlaid. Such deformities are important because they can result in disks where the blades do not fit properly into the dovetail slots.

Additionally, certain regions of the post, called pressure faces, are of particular importance because these are the areas against which the blades press when the engine is running. Due to high stresses during operation, any deformation in these areas needs to be carefully measured.

![Figure 2](image1)

There are several competing requirements for a system of algorithms that addresses these issues: it must be accurate, robust, and fast. A post deformed by as little as 5 mils is considered defective. Further, if the system produces false alarms or the computation time takes more than the approximate 5 seconds that it takes a user to inspect a post, the user will lose patience or confidence in the system and will likely turn it off. There is a natural tradeoff between accuracy and speed: to increase accuracy, we can, for instance, increase the resolution of the cameras, but this results in larger images that take longer to process.

To address these requirements, we developed a system that detects these minute (less than 0.005 inches) post deformations in less than 4 seconds. The system first creates a “gold-standard” template post that has normal contours and then compares the contour of this post to the contour of each new post to measure the deformity. It gives a warning to the user if the deformation is above a certain amount and outlines the location of the deformation.

Section II describes this system. The steps consist of offline processes that are done during a training phase and online processes that are done for new posts as they are acquired. Section III presents the results of applying the system to analyze a large number of posts. This section gives comparisons between the measurements on normal posts and deformed posts. Finally, Section IV discusses the conclusions of the work and points out areas for future work.

II. METHODS

A flowchart of the processing is given in Figure 4. This flowchart consists of offline and online processes. In the offline processing, the gold-standard template is created, and the contour of the template is found along with the template contour indices corresponding to the pressure faces. These measurements are calculated once and are saved for use in the online processing. In the online processing, a new post image is acquired and compared with the template post. The details of these steps are described in the rest of this section.
A. Offline Processing

The main purpose of the offline processing step is to generate a gold-standard template image that can be compared with images of new posts. This step is done once, offline, so the emphasis is on the accuracy of the template creation rather than the speed of the algorithm.

1) Smoothing: The images in both the offline and online steps are first smoothed in order to remove spurious noise while retaining the edges. The system includes three possible smoothing algorithms: Gaussian smoothing, median filtering, and a curvature-based level set approach called the modified curvature diffusion equation (MCDE) described in [4].

2) Registration: Both the offline and online processing steps require an image registration step. Registration is an optimization problem of finding the transformation that brings the two images into alignment. The registration is performed using the framework of the Insight Toolkit (ITK) [5], which is a C++ based, open-source, cross-platform toolkit for performing registration and segmentation. This framework consists of several elements that can be interchanged.

Figure 5 shows the various elements of the framework. In this case, the fixed image is the template image and the moving image is the image of the new post. The transform component represents the spatial mapping of points from the fixed image space to points in the moving image space. The major mis-registration of the posts is due to the rotation of the disk, leading to a rigid transformation consisting of a translational and rotational component. The interpolator is used to evaluate moving image intensities at non-grid positions. We used a linear interpolation scheme for speed. The metric provides a measure of how well the fixed image is matched by the transformed moving image. We used a mutual information metric because the posts are taken under various illumination conditions. The optimizer optimizes the quantitative criterion of the metric over the search space defined by the parameters of the transform. We used a gradient descent optimizer. [5] In addition, we used a multi-resolution pyramid registration process to speed the registration, whereby the images are first registered at a coarse scale, and successive refinements are made to the transform as the registration moves to finer scales.

3) Template Generation: An exemplar set of nominal post images is collected for use in generating a template image. Once the images have been smoothed, a template image is automatically generated from the set of images by iteratively averaging all of the images registered to one of the images. All of these registered images are averaged, and this generates a template. In the subsequent iterations, the template created by averaging the registered posts from the previous iteration is used to re-register the images.

The registration algorithm calculates a transform between the two images and uses this transform to resample the moving image. Therefore, some pixels on the border of the output image do not come from any pixel in the moving image, and these pixels are set to 0. When these images are simply averaged together using

$$\mu_{i,j} = \frac{1}{K} \sum_{k=1}^{K} I_{i,j,k} \tag{1}$$

(where \(I\) is the image intensity, \(i\) and \(j\) indicate image row and column indices, \(k\) indicates the image number, and \(K\) is the number of images), these “invalid” regions result in streaks along the four borders of the image where “invalid” pixels are averaged with “valid” pixels (see the top image of Figure 6). This can cause problems during the registration step. To overcome this, we developed a method of selectively averaging the image. This is given by

$$\mu_{i,j} = \frac{1}{S_{i,j}} \sum_{k=1}^{K} I_{i,j,k} M_{i,j,k} \tag{2}$$

where \(M\) indicates the binary mask resulting from transforming an image of ones using the same transform as the registered image. This mask is unity where the pixels are “valid” (i.e. where a transformed pixel of the moving image is used) and zero otherwise. \(S_{i,j}\) indicates the number of “valid” pixels across all images for a given
image position. It is found from the mask images $M$ as

$$S_{i,j} = \sum_{k=1}^{K} M_{i,j,k}$$  \hspace{1cm} (3)

The result is shown in the bottom image of Figure 6. Notice that the streaks are no longer present.

Figure 6. Average and selective average templates. The corruption of the “invalid” pixels can be seen as dark vertical and horizontal streaks along the left, top, and bottom borders in a). The template image found by selective averaging in b) is much cleaner and yields better registration results.

4) **Template Contour Creation**: The template contour is also needed for the online processing. It is automatically created from the template using an edge detector followed by morphological operators to find only the edges corresponding to the contour of the post, which is where the deformation occurs. The coordinates of these contour points are used in the online processing stage to compare with the edges of new posts.

A subset of these contour points are of particular interest since they correspond with the areas of the pressure faces. Figure 7 shows the template image with the template contour points corresponding to pressure faces color-coded and those not corresponding to any pressure faces are colored white.

Figure 7. Template contour points and pressure faces. The template image is shown with the template contour overlayed. The contour points corresponding to pressure faces are color-coded, and those not corresponding to any pressure faces are colored white.

areas can be found. The deformation at these pressure faces can then be calculated in the online processing stage.

B. **Online Processing**

The online processing takes place for every new post image that is acquired. This step uses many of the same steps as the offline processing such as image smoothing, registration, and edge detection. The resulting post contour is eventually compared with that of the gold-standard template to define a deformation score. In contrast with the offline processing stage, however, in this stage, speed is essential since the user must wait for the results of the measurement while the processing is carried out for each post. Since accuracy is also essential, we have developed several methods to speed up the algorithm while retaining a high level of accuracy. These optimizations as well as efficient pixelated and sub-pixel contour matching are described in this section.

1) **Algorithm optimizations**: A resampling is first performed on the image, which speeds the algorithm by reducing the number of pixels necessary for the computation. We use bilinear interpolation for this stage, where resampled points are interpolated based on

$$f_{x,y} = \begin{bmatrix} 1 - x_d & x_d \\ y_d & 1 - y_d \end{bmatrix} \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} 1 - y_d \\ y_d \end{bmatrix}$$  \hspace{1cm} (4)

$$x_d = x - \lfloor x \rfloor, \quad y_d = y - \lfloor y \rfloor$$

Processing time is also reduced by cropping the image around the post before registration. Only a region of interest surrounding the post is registered rather than the entire image. This speeds the registration step by providing an initial transform that closely aligns the posts. It also speeds the registration and subsequent steps by yielding fewer pixels on which to operate. Cropping is accomplished by first thresholding the gradient magnitude image to retain the pixels corresponding to the edges.
of the post and then cropping the image to the box bounded by these edge points. It is not possible to hard-code the threshold value because of varying illumination conditions. Therefore, we calculate it dynamically for each new post based on the sorted pixel value of the gradient magnitude image as

\[g_0 \leq g_i \leq g_N\]

\[t = g[p|N]\]

(5)

where \(g_i\) are sorted gradient magnitude pixels, \(p\) is a fraction between 0 and 1, and \(N\) is the number of pixels in the image. Typically, we set \(p = 0.95\), and the majority of edge pixels are in this category.

The posts are then registered to the template image. One method of speeding up the computation is to register only a subset of the image pixels using a random sampler with a set number of samples. In addition, an initial transformation based on first moment calculations that find the approximate post centers can be used. Moments of image intensities are often calculated for this purpose, and this works well for images where the surrounding background is dark. However, the posts are generally bright inside the object and bright in the background with varying contrast, so this is not effective. Instead, we calculate the first moment of the gradient magnitude of the image as in Equation 7 since the gradient magnitude has largest values at edges, leading to a centroid that is close to the center of the post.

\[M_{i,j} = \sum_{y=0}^{Y-1} \sum_{x=0}^{X-1} x^i y^j I(x,y)\]

(6)

\[\text{Centroid} : \begin{bmatrix} M_{1,0} & M_{0,1} \\ M_{0,0} & M_{0,0} \end{bmatrix}\]

(7)

2) Distance Calculation: After the new post image has been registered to the template, the difference in the contours of the posts needs to be measured. Computing the distance between two contours can be an expensive operation. We simplify the calculation by finding a distance map from the new post contour pixels and then sum the distances at the locations where the template pixels fall. To accomplish this, we first find the Canny edges [6] of the new post. The Canny algorithm requires the setting of several parameters including the upper and lower thresholds used for hysteresis thresholding. As in the cropping step, it is not possible to hard-code these values because of varying illumination conditions, so we calculate them for each new post based on Equation 5, where we set \(p = 0.95\) for the upper threshold and \(p = 0.6\) for the lower threshold.

Given the resulting edge image of the new post, the minimum distance from every other image point to pixels on this edge is calculated using the fast Danielsson algorithm [7]. This enables the computation of the Chamfer distance [8] at every point of the template contour. This gives, for each template contour point, the distance to the closest edge of the new post. A measure of the deformation distance of the post can then be found by averaging these values across the entire contour as

\[\bar{d} = \frac{1}{N} \sum_{i=1}^{N} d_i\]

(8)

where \(N\) is the total number of contour points, and \(d_i\) is the distance at a template contour point \(i\). Additionally, this average distance can be calculated for each of the pressure face regions, resulting in 6 localized measures of the deformation distance. These 7 distances can then be compared to reasonable thresholds to determine whether the post is normal or deformed.

3) Sub-pixel Distance Calculation: The specifications of the problem require a high level of precision in the deformation calculation. It is therefore desirable to consider sub-pixel distance calculation as a means of improving the accuracy of the system. For example, Figure 8 shows a comparison between a pixelated and sub-pixel detection of a post contour, demonstrating the greater smoothness and accuracy of the sub-pixel contour. However, the calculation of the sub-pixel distance between two curves is generally a computationally intensive problem. A brute-force method would search all points on each curve to find the correspondence that minimizes the distance between points. If the number of points on one curve is \(N\), and the number of points on the other curve is \(M\), this amounts to \(NM\) computations, where \(N\) and \(M\) are both on the order of approximately 3000 in our case. Another standard method for computing the distance is to project a ray normal to one curve and find where it intersects the other curve. This is also computationally expensive because it requires fitting of curve points to find the normal and searching to find the closest point on the second curve.

In [9], Kimmel et. al. present a method for generation of sub-pixel distance transforms, and Frenkel and Basri in [10] present a method for using the Fast Marching algorithm for matching curves. However, computational efficiency remains an issue.

We have developed an efficient algorithm to perform this calculation using the result of the pixelated distance map. To find the sub-pixel distance, it is necessary to find the pixel on the new post contour to which each template contour pixel is closest. If this correspondence can be found, then the sub-pixel distance can be calculated as the distance between the sub-pixel point at the template contour pixel and the sub-pixel point at the new post contour pixel. This is detailed in Algorithm 1.

The only step that is not defined in Algorithm 1 is that of mapping the distance found from the distance map at a template contour pixel to the possible pixel indices. The pixel offset indices give the offset in pixels from the template contour pixel to the closest new post contour pixel.

Several observations converge to yield a computationally efficient algorithm for this step. First, if a circle were drawn around a template contour pixel with a distance equal to the value of the distance map at that pixel, the edge of this circle would intersect at least one pixel of the
Algorithm 1 Efficient algorithm for calculating the sub-pixel distance between two curves.

1: for Every template contour pixel \( t \) do
2:   Find the pixelated distance to the new post contour using the distance map.
3:   Find the mapping of the distance to the candidate pixel offsets.
4:   Set sub-pixel distance \( d_s = \infty \).
5:   for Every candidate pixel offset \( c \) do
6:     if The distance map at the pixel offset is 0 (meaning it is on the new post contour) then
7:       Find sub-pixel template contour point \((x_t,y_t)\).
8:       Find sub-pixel new post contour point \((x_c,y_c)\).
9:       Find sub-pixel distance for this candidate \( d_c = \sqrt{(x_c - x_t)^2 + (y_c - y_t)^2} \).
10:      if \( d_c < d_s \) then
11:         \( d_s = d_c \).
12:     end if
13:   end if
14: end for
15: end for

Figure 8. Comparison between pixelated and sub-pixel contour detection. The sub-pixel contour leads to a much smoother and accurate transition than the pixelated contour.

Figure 9. Efficient candidate points example. The graph shows all possible pixel locations that have a distance of exactly 5 from the origin. Of all of the pixels on the circle surrounding the origin, these are the only possible pixels that are a distance of exactly 5 using a pixelated distance map. The points marked in red represent the values stored in the mapping from the distance, 5, to the possible pairs of integer values \((0,5),(5,0),(3,4),(4,3)\). Only the positive values are stored for efficiency. Upon query, the other positive and negative permutations, shown in blue, can be easily generated.

new post contour. Another constraint is that the circle has to intersect the closest new post contour pixel exactly in the center of the pixel. This is true because otherwise the distance would be different. This is equivalent to stating that the \((x,y)\) offset from the template contour pixel to the closest new post contour pixel must consist only of integers.

For example, consider the distance \( d = 5 = \sqrt{x^2 + y^2} \). The possible values of \( x \) and \( y \) are found as \( d^2 = 5^2 = 0^2 + 5^2 = 5^2 + 0^2 = 3^2 + 4^2 = 4^2 + 3^2 \) as well as the positive and negative permutations thereof. This is illustrated in Figure 9 where the red points indicate the indices listed above, and the positive and negative permutations are in blue. Note that while the offsets \( x \) and \( y \) must be integers, the distance need not be an integer. For example, if \( x = 2 \), and \( y = 5 \), the distance will be \( d = \sqrt{2^2 + 5^2} \approx 5.385 \). To avoid round-off error, we generally work with \( d^2 \) instead of \( d \). Figure 10 gives an example of the overall sub-pixel distance calculation described in Algorithm 1.

To realize this algorithm, the mapping is generated at run-time according to Algorithm 2. The algorithm first checks whether the mapping has already been generated. If so, the algorithm returns with the mapping. Otherwise, the value for \( x \) is solved while \( y \) is swept in integer steps from 0 to floor\((d)\). When considering only positive values for \( x \) and \( y \), \( 0 \leq y \leq d \) by definition. For each \( y \) value where \( x \) is also an integer, this mapping is added as a candidate. This generates all results where \( x \) and \( y \) are both positive. The algorithm returns with the possible mappings found in this way, and the positive and negative permutations can be easily generated therefrom.

The complexity of this algorithm is only \( O(D^3) \), where \( D \) is the maximum \( d \) for all template contour pixels. To demonstrate this, a matrix can be constructed where the \( x \) and \( y \) indices go from 1 to \( D \), and the entries represent the sum of the squared distances \( d^2 = x^2 + y^2 \). There are
Algorithm 2 Algorithm for mapping from distance to candidate offset indices
1: if \( d \) mapping has already been found then
2: return mapping
3: else
4: for \( y = 0 \) to \( |d| \) do
5: \( x = \sqrt{d^2 - y^2} \)
6: if \( x \) is an integer then
7: Add candidate offset pair \((x,y)\).
8: end if
9: end for
10: return mapping
11: end if

III. RESULTS

Because of the limited availability of production post images with real defects, an engineered defect post was created that accurately mimics the types of deformations that appear in these posts. Of the 80 posts on the disk, 79 are normal, and one was deformed by 5 mils by re-broaching the part. The “Aid to Visual” system was used to acquire images of size 1280*960 of all of the posts, and these images were then passed into the deformation detection system. For this resolution, the pixel size is 1 mil on each side.

A. Repeatability Tests

To determine the repeatability of the system to detect deformations on the same post, we recorded 10 images of the deformed post in succession. We moved the stage forward and backward in between each image capture. This resulted in 10 images of the same post transformed by various translations and rotations in the image. The resulting measurements of the system are summarized in Table I, which shows that the system is able to consistently measure the deformation with high accuracy.

<table>
<thead>
<tr>
<th>Image Number</th>
<th>Pixel Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.874</td>
</tr>
<tr>
<td>2</td>
<td>1.889</td>
</tr>
<tr>
<td>3</td>
<td>1.853</td>
</tr>
<tr>
<td>4</td>
<td>1.816</td>
</tr>
<tr>
<td>5</td>
<td>1.814</td>
</tr>
<tr>
<td>6</td>
<td>1.804</td>
</tr>
<tr>
<td>7</td>
<td>1.846</td>
</tr>
<tr>
<td>8</td>
<td>1.851</td>
</tr>
<tr>
<td>9</td>
<td>1.827</td>
</tr>
<tr>
<td>10</td>
<td>1.827</td>
</tr>
</tbody>
</table>

Mean: 1.840
Std: 0.027

B. Robustness and Timing Tests

We collected four independent sets of images of the disk. Each set consisted of acquiring images while rotating the disk through the system. The results can be seen in Figure 11, which shows each test in a different color. The x-axis represents the post number, and the y-axis represents the deformation distance. It can be clearly seen that, in all four tests, the deformed post (post #17) stood out as having a much larger distance.

Although the results of Figure 11 clearly show a distinction between the normal posts and the deformed post, some accuracy is lost in averaging the distances around the entire post contour, especially since only one side of the post was deformed. The pressure faces distances give larger distinction because of their localized nature. The results for the six pressure faces are given in

Figure 10. Synthetic sub-pixel distance example. The white dots represent the sub-pixel locations of the contour of the new post, and the 0 inside of the pixels represent that these pixels have a distance of zero from the contour. The colors represent increasing distances from the contour. The x-axis represents the deformation distance. It can be clearly seen that, in all four tests, the deformed post (post #17) stood out as having a much larger distance.

0.027
Figure 11. Distance results. Overall distances in mils for all 80 posts of all four datasets are shown. The deformed post clearly stands out above the others. The fluctuations in the measurements are due to slight post variations and partial voluming effects.

Figure 12. Pressure face distance results. Distances in mils for the pressure faces of all 80 posts of the first dataset are shown. All six pressure face measurements are shown for each post. The pressure face distances for the deformed post are on the right side that was rebroached (the three largest peaks in the graph) and low on the other side.

Figure 12 for the first of the four tests. Here it can be seen that the deformed post (#17) sticks out even more prominently. It can also be seen that the distances for pressure faces 4-6 are much greater than those of 1-3, where the numbering corresponds with that in Figure 7. Since the pressure faces are numbered clockwise from the top left, the last three correspond with the side of the post where the deformation exists. Thus, the distances of the last three pressure faces are more indicative of the true deformation, and, using these measurements, it should be possible to identify posts that are deformed as little as 2 mils. These results were almost identical for the other three datasets. Note that the threshold chosen to identify a pressure face deformation should be higher than that for the overall deformation since the averaging effect of the overall deformation tends to decrease the variability.

The average processing times for the four datasets are 3.75, 3.75, 3.84, and 3.74 seconds on a standard laptop with a 1.86 GHz processor with 2GB of RAM. These times are the averages for the 80 posts in each dataset. These times demonstrate that the system is able to operate on average less than 3.8 seconds, which is a sufficiently rapid rate that it will not interfere with the work-flow of the inspector.

C. Sub-pixel Results

To validate the improvements in accuracy achieved by the sub-pixel distance calculation over the pixelated distance calculation, we generated a synthetic image of size 512*512 consisting of a white circle with radius 120 on a black background. We then applied a transform with different amounts of scaling: 0.5, 2/3, 3/2, and 2 (where the scaling value indicates by how much the radius is divided). This generated interpolated images with circles of various sizes. With an original radius of 120, these scaling values result in a radius change (or, equivalently, the distance to the contour $d_{true}$) of $|120 - 120| = 120$, $|120 - 120(2/3)| = 60$, $|120 - 120(3/2)| = 40$, and $|120 - 120(2)| = 60$, respectively. For each of these transforms, we resampled the original image (template) and new image by 1, 0.75, 0.5, and 0.25. In each case, the blue bar represents the pixelated error and the purple bar represents the sub-pixel error. The sub-pixel error is substantially lower than the pixelated for almost all cases.

D. Large Scale Validation

For validation purposes, the algorithms were tested on a set of images from 30 experiments, each with 80
posts, representing 2400 posts acquired using the Aid To Visual system. The images were gathered at two time points, 6 months apart. The first time point included 21 experiments, and the second included 9. All of these posts are known to be normal, without deformations.

During the inspection, the ring of posts is often turned over and imaged on the opposite side so that the same camera can take images of the opposite side of the post. We refer to the two configurations of the posts as shaft_up and shaft_down, and typically only one of these views is taken for each experiment. While the views are similar, the parts look different enough that they require separate templates. Out of the 21 experiments of the first time point, 15 were taken with shaft_up, and 6 with shaft_down, and out of the 9 experiments of the second time point, 4 were taken with shaft_up and 5 were taken with shaft_down.

To classify the performance of the system, we have set a threshold based on the deformation score of the deformed posts described in Section III-A. According to Table I, the average deformation score is \( \mu = 1.84 \), and the standard deviation \( \sigma \) is 0.027. Using these results, we set the threshold to be \( t = \mu - 5\sigma \approx 1.7 \). Any post that exceeds this threshold is considered to be a deformed post; otherwise, it is considered to be normal. The results for the 2400 posts are given in Table II. Since all posts are known to be normal, any post classified as deformed is counted as a false positive.

Several conclusions can be drawn from Table II. First, the algorithms perform very well, with an average error rate of less than 1%. Secondly, it is clear that Group 2 performs approximately twice as well as Group 1. This is mostly due to the improved contrast of the images. Between the acquisition of the two groups, the lighting conditions were modified to improve the contrast. Figure 14 demonstrates this improvement in the lighting conditions by showing an example image from each of the groups. As the contrast improves, the edges are more clearly defined, and the algorithms are able to more accurately compare the template contour with that of the new post. This suggests that further improvements of the acquisition conditions will lead to even greater reductions in the algorithm errors.

The disk positioning control system in the ATV system drifts over time which results in posts that are generally not centered in the camera field of view. If the post moves out of the field of view (see Figure 15), the system is manually re-centered. In this case, the algorithms warn the user by reporting that the part is deformed since it technically is missing some pieces. For the results in Table II, the images of the posts taken under these conditions were treated as deformed, and they did not contribute to the error score. One of the topics for future work involves using the registration transform to provide feedback to the stepper motor to correct itself if the post starts to move out of the field of view.

### E. Parameter Optimization

We performed a Design of Experiment (DOE) to optimize the large number of parameters to ensure that the system can distinguish all of the normal posts from the deformed posts for each experiment. Since we are interested in the distance of the deformed post \( x_i \), the distances of the normal posts \( X \), and the standard deviation of the normal post distances, we used the Z score as a single summary statistic for the accuracy of a complete experiment, where

\[
Z = \frac{x_i - \text{mean}(X)}{\text{std}(X)}
\]

The use of this Z score metric provided for an efficient way to optimize the algorithm parameters and systematically address the inherent tradeoffs.
The parameters of the system and their selected values are summarized in Table III. They can be broken into three categories: smoothing, registration, and algorithm speed-up. In the smoothing category, three types were considered: Gaussian smoothing, median filtering, and the level set based modified curvature diffusion equation (MCDE) smoothing. In the final analysis, the time requirements dictated that the Gaussian filter be used even though the MCDE yielded the best edge localization results. The optimal parameter settings for each of these algorithms is given in the table, and the “smoothing type” indicates that the “Gaussian” method was chosen.

The second set of parameters are related to the registration algorithm. The “number of levels” is the number of resolution levels in the pyramid. The “number of iterations” is the maximum number of iterations for each pyramid level of the registration. At each iteration, the optimizer will take a step along the direction of the derivative; “max step length” indicates the initial length of the step. Each time the direction of the derivative abruptly changes, the optimizer assumes that a local extrema has been passed and reacts by reducing the step length by a fraction; this is the “relaxation factor”. After several reductions of the step length, the optimizer may be moving in a very restricted area of the transform parameter space. The “min step length” defines how small the step length should be to consider convergence to have been reached. The “step length factor” is the factor by which the minimum step length is reduced at each level of the pyramid in order to allow the optimizer to focus on progressively smaller regions. The “gradient magnitude tolerance” defines a stopping criterion based on the amount of change of the transform from one iteration to the next. For the transformation, the units in rotation and translation are quite different, and we can define scaling weights for these transformations. In general, the posts can be translated a great deal whereas the rotation is generally slight. The “translation penalty” is therefore small whereas the “rotation penalty” is much higher. The “center penalty” refers to scaling of the center of the rotation.

The third set of parameters are related to the algorithm speed-up. The “scale factor” defines the amount of re-sampling of the entire image. The boolean “crop image” determines whether to crop the image. The “use moment calculation” determines whether to use the moment calculation to initialize the registration. If the cropping is used, it is not necessary to also use the moment calculation since the initial registration location will already be set. The “fraction of samples” defines what fraction of the pixels in the image should be used for the registration. This is part of the random pixel sampler described in Section II-B.

### F. Integration into LabVIEW

The algorithms described in this paper have been implemented as a usable system for inspectors. The Aid to Visual system uses LabVIEW, a registered trademark of National Instruments, for acquiring and processing the images. The system described in this paper was integrated into LabVIEW by generating a single dynamically linked library (DLL) of the algorithms that is called using the LabVIEW interface and returns the Chamfer distance for the entire post as well as the pressure faces. It compares these distances to thresholds and gives a warning to the user if the distances are above the thresholds.

The inverse of the transform calculated by the registration algorithm is used to transform the template contour points back into the coordinate frame of the new post so that they can be displayed on top of the post if the thresholds are exceeded. This provides visual feedback to help the user identify deformed areas of the post.

A close-up of the LabVIEW diagram showing the algorithm module is in Figure 16. The inputs on the left are as follows, from top to bottom. Three blue bracket inputs: template image, new post image, registered image (this is a pointer that is filled by the algorithm). Two blue “32” inputs: image size (length and width). Four orange “SGL” inputs: image spacing (x,y) and origin (x,y). “abc” input: name of the parameter file where all the parameter values are stored. One orange bracket input: empty 7-element array of Chamfer distances (overall distance and pressure face distances). Another orange bracket input: n-element array of contour points, where n is the number of points. Blue “Ul-b” input: flag for whether to write out binary masks for selective averaging. The outputs on the right are as follows, from top to bottom. First orange bracket output: 7-element array of Chamfer distances. These distances are then thresholded and fed to a module that raises a flag if the distance is too high. Second orange bracket output: template points transformed to the coordinates of the new post. These transformed points can be overlaid on the post as a contour if the post is deformed.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Name</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Smoothing</strong></td>
<td>smoothing type</td>
<td>Gaussian</td>
</tr>
<tr>
<td></td>
<td>Gaussian sigma</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>median radius</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>MCDE iterations</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>MCDE time step</td>
<td>0.125</td>
</tr>
<tr>
<td></td>
<td>MCDE conductance</td>
<td>20</td>
</tr>
<tr>
<td><strong>Registration</strong></td>
<td>number of levels</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>number of iterations</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>max step length</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>relaxation factor</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>min step length</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>step length factor</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>gradient magnitude tolerance</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>translation penalty</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>rotation penalty</td>
<td>1000000</td>
</tr>
<tr>
<td></td>
<td>center penalty</td>
<td>1000000</td>
</tr>
<tr>
<td><strong>Speed up</strong></td>
<td>scale factor</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>crop image</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>use moment calculation</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>fraction of samples</td>
<td>0.2</td>
</tr>
</tbody>
</table>
IV. CONCLUSIONS AND FUTURE WORK

We presented an Aid to Visual system that can effectively detect minute deformations in engine turbine posts, which are extremely difficult to detect even for trained inspectors. New posts are registered to a reference template and the difference between the reference contour and the test post is computed. We presented several methods to address the inherent tradeoff between speed and accuracy in order to meet the demanding requirements of this application. Computationally efficient pixelated and sub-pixel contour matching algorithms are introduced that require only a small fraction of a second to compute. Results on over 2400 posts demonstrate a false positive rate less than 1%. This system has been integrated into LabVIEW and is being deployed as part of a functioning inspection system. Future work involves both analysis of how the amount of rotation and translation affects the detection results and continued monitoring of the system to identify potential areas for improvement.

ACKNOWLEDGMENTS

We thank Joe Ross and David Raines for collecting and transferring the data and for clarifying the requirements of the system. We also thank the reviewers for their detailed and helpful comments.

REFERENCES


Dirk Padfield

Dirk Padfield earned undergraduate degrees in electrical engineering and international studies from the Pennsylvania State University in 2000. He earned the Master of Science in electrical engineering from the same institution in 2002. He is currently working on his Ph.D. in computer science at the Rensselaer Polytechnic Institute. He worked as a research assistant for NASA in 1999 and 2000, for the National Science Foundation from 2000-2002, and for the National Science Council of Taiwan in 2002. He is currently a computer scientist at the GE Global Research Center in Niskayuna, NY. His research interests include image segmentation, registration, and cellular image analysis. He has served as a reviewer for the IEEE TBM and IEEE TITB journals and the MICCAI, ACCV, ISBI, EMBC, and MIAAB conferences.

Glen Brooksby

Glen Brooksby received an undergraduate degree in electrical engineering from Brigham Young University in 1987 and an M.Eng. from Rensselaer Polytechnic Institute in 1994. He joined GE Global Research in 1989 as an Electrical Engineer and has been involved in a variety of research projects in the areas of communications, optics, and computer vision with applications in digital satellite communications, voice coding, and, most recently, industrial inspection and aerial image analysis. Prior to working for GE, he worked for Ford Aerospace Corp., Space Systems Division in the RF Equipment Engineering group from 1987 to 1989. Mr. Brooksby currently holds 21 issued patents.

Robert Kaucic

Robert Kaucic received his BS in Computer Science and Electrical Engineering from the US Air Force Academy in 1985, his MSEE from the University of Washington in 1987, and his D.Phil. from Oxford University in 1997. He served in the US Air Force from 1985-1999 where he attained the rank of Lieutenant Colonel. In October 1999, he joined General Electric Global Research where he is currently a research scientist. He leads a global group of engineers and scientists developing computer vision solutions to industrial imaging problems including manufacturing inspection and anomaly detection. He has published across the computer-vision landscape to include video object tracking, scene reconstruction, self calibration, texture classification, image segmentation, medical image analysis, and industrial inspection. He is a member of IEEE and the PAMI technical committee.