Automating Critical Surface Identification and Damage Detection Using Deep Learning and Perspective Projection Methods

Gautam Kumar Vadisala[1], Anurag Singh Rawat[2], Abhishek Dubey[3], Gareth Yen Ket Chin[4] and Fabio Abreu[5]

[1,2,3]*Schlumberger, Building 8, Office 301, Commerce zone IT Park, Pune, Maharashtra-411006, India*

GVadisala@slb.com
ARawat4@slb.com
ADubey4@slb.com

[4,5] *Schlumberger Technology Corporation, Rosharon Testing and Subsea Center, Rosharon, Texas-77583, US*

GSchin@slb.com
FAreu3@slb.com

ABSTRACT

With an increased collection of data, assessing the health of an asset and designing recommendations or executing response actions via prognostics and health management (PHM) has made great advances. These actions can be corrective or preventive depending upon the risk of failure or the cost of repair. As downhole testing tools operate in extreme environments, they are subjected to conditions like elevated temperature, shocks, vibrations, and pressures. The dump mandrels used in the process are prone to wear and tear like scratches, pits, and corrosion, which may cause operational failure. If these damages and their degree goes undetected and no remedial actions are taken, possibilities of non-productive time (NPT) and financial losses increase drastically. This paper aims to develop a fitness inspector which uses Computer Vision and Deep Learning to identify critical surfaces of these tools and the damage within them. This will help the Subject Matter Experts (SMEs) by replacing the qualified workforce provided by them and reducing the time consumed to gauge the health status of all the tools as the diagnosis can be made in real-time.

1. INTRODUCTION

Health management is an important aspect of tool lifecycle management. With correct management of data, we can have full visibility into the health of an asset throughout its lifecycle, from design to production to obsolescence. We have access to the past data of the whole fleet to analyze any anomaly and diagnose why it happened. By connecting the learnings from past data to the real-time data coming from an asset, we can get a current health assessment and diagnose the root cause of an anomaly as it is happening.

After operating in fields, Down Hole Testing tools such as dump mandrels have damage on the sealing surfaces and O-ring grooves. This mandates a tool inspection by an SME after every use to decide whether it would be fit for the next job. This process invites two types of risks which are, scrapping good tools too early and using a questionable tool in a million-dollar worth of field operation. Also, this would require a lot of effort by subject matter experts (SMEs) in terms of the inspection time. We aim to reduce this inspection time and provide a data-based decision-making process.

We aim to propose a real-time visual platform during an inspection, that would take a video of the tool being inspected as an input and provide the decision of whether the tool can be used for the next operation or to be sent for repair. This video can be taken from a Digital single-lens reflex (DSLR) camera or a mobile phone in a fixed setup. Although, we felt that the former would provide better results (this is explained in the further sections of the paper).

As dump mandrels are cylindrical, we need to record the video of the mandrel by rotating it along its axis to capture all its surface area for inspection. The video is then further processed into individual frames for analysis. In each frame, we first detect a reference point through which we locate the critical surfaces of the tool and then identify the damage on those surfaces. We also need the engineering designs of the mandrel to locate the critical surfaces or the area of interest in the frames. We perform these steps on all frames and produce the final result.

We consider the center of the mandrel edge as a reference point as the critical surface areas are mentioned from this edge in the engineering designs. To detect this point, we compute the mandrel boundaries in each frame by a Deep Learning based Object Detection method. After locating the edge of the mandrel in the video frame, we project the critical surfaces on it by estimating the distance of the critical surface in terms of pixels. We consider the diameter of the tool as a reference measure throughout the video and use it to convert the actual distance in inches (or mm) to pixels. As every prediction result from the model is probabilistic, this causes dislocation of critical surfaces from their actual location, sometimes by a large degree in the resultant video if the input video was shaky and the rotational speed of the video was not constant. Another major issue we encountered was the conical projections of the tool in the video. As the mandrel is cylindrical, it is difficult to precisely follow the surface at different points by this method. So, this deep-learning-based method is not able to address this issue as we can only project vertical lines for critical surfaces given their distance from the reference point.

To solve these issues and to make the projection of critical surfaces similar throughout the video, we went for a fixed apparatus where the camera would be placed at a fixed distance and the tool would be rotated at a constant rotational speed. We can then leverage Camera Projection methods to convert the real-life distances in inches (or mm) to the distance in pixels. This would also take care of the curvilinear surfaces as our projections follow the 3D nature of the tool in the video.

For damage detection on the tool surface, we train a Deep Learning based Semantic Segmentation model based on U-net architecture. As the surface damages would be small and different, image-processing-based semantic segmentation methods are not helpful. Also, this model only requires a few images for a good-enough solution and can be improved with more data. After detection of damage, we project the damage identified on the critical surfaces or our area of interest.

After the projection of damage on the critical surfaces, we can leave it to the SME to decide on further actions or we can create a metric to compute the percentage of damaged pixels from the whole critical surface areas in pixels. By using this methodology, we can provide an automated solution for end-to-end tool inspection. Our method can be easily replicated for other tools with known geometry to have a surface-level inspection.

2. RELATED WORK

Cylindrical objects like Mandrels tend to have perspective effects when captured in an image. This means the points on the cylindrical surface which are at some distance from the straight line of vision of the camera will appear in curves on the image. For example, points on the left side of the camera will appear curved and their focus will be on the camera center. A solution for this problem of cylindrical objects producing perspective effects like conical sections in images was previously presented in the article (Berveglieri & Tommaselli, 2018). Their method aims at performing a continuous reconstruction of 3D cylindrical patches with high accuracy. They collect the images at different viewpoints and then fit the corresponding image patches using a modified geometric transformation for Least Square Matching (LSM) (Gruen, 1996). The image acquisition is performed by displacing the camera in a line path parallel to the cylindrical axis, as shown in the below figure (Figure 1).

In Figure 1, three views of a strip over the cylinder that correspond to the strip over the cylinder are displayed. The strip appears as a horizontal shape (the main axis appears as a straight line) when the image is collected from a normal view (the projecting ray to the strip over the cylinder). If the projecting ray of the strip path is oblique, then the strip has a curved shape. This occurs due to the cylindrical shape of the object and the change in camera viewpoint. Given two image patches or regions that refer to the same area over the cylindrical diameter but with different perspective views, parallaxes will occur due to depth and orientation variations.

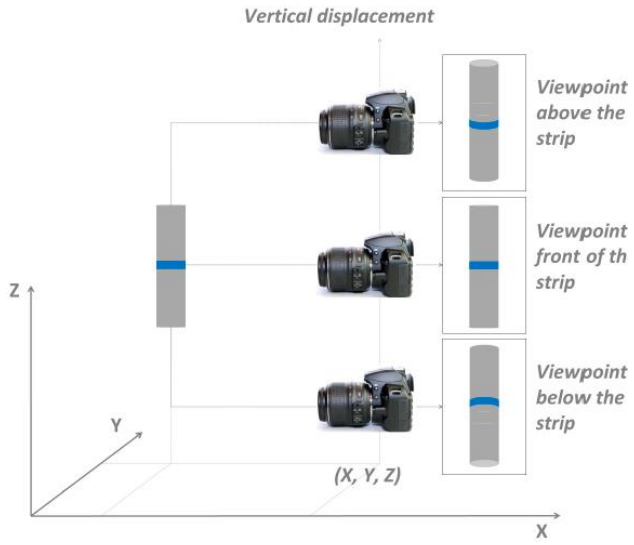


Figure 1. Image acquisition using camera displacement (Berveglieri & Tommaselli, 2018)

The method proposed by Berveglieri and Tommaselli (2018) aims to reconstruct the 3D cylindrical surface by using a geometric transformation T (as mentioned in Figure 2) with further refinement by an adaptive least square matching (ALSM) (Gruen, 1985), to accurately map a point from an image $I_1(x, y)$ to its respective correspondence in an image $I_2(x, y)$ with sub-pixel precision. Although the above method provides a solution to the conical sections in images, we are looking for a solution without the surface reconstruction. Also, this method brings up the data requirement of collecting the videos of a mandrel across different viewpoints. So, we

propose a method where we use Camera Projection techniques and project a real-world 3D object onto a 2D image. Also, our solution only requires one video of the mandrel.

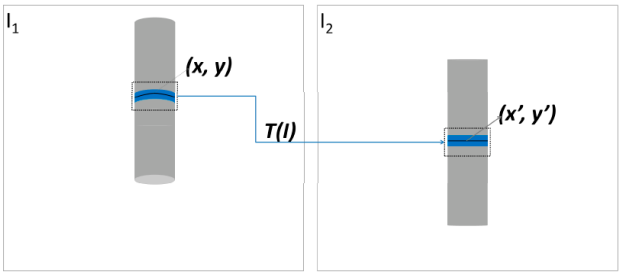


Figure 2. Matching using Least Squares (Berveglieri & Tommaselli, 2018)

3. METHODOLOGY

The proposed solution has three stepped approaches. First, the reference point is identified in the video frame. Second, the area of interest i.e., the critical surface areas which are of concern are identified in the same frame. Then the damage is detected within the critical surface on the mandrel in that frame. For final decision making, we combine the above solution for all the video frames and consider the damage within the critical surfaces.

3.1. Reference Point Identification

To identify critical surfaces on a mandrel, we first need a reference point from which we can project them on the mandrel in the image. Since the dump mandrel is cylindrical, we felt the center of the edge would be suitable for this purpose. But to find this point, we need to detect the mandrel boundary in the image. There are many methods from which we can estimate the mandrel boundaries, but from our observations, edge detection methods give approximate results for computing the object boundary.

3.1.1. Edge Detection

The above time constraint prompted us to look for a solution that would be easy to compute and in a lesser amount of time. We felt that if the mandrel is in the foreground and the video is recorded with a clear background and without much distortion, we can employ edge detection methods such as Canny Edge Detection (Canny, 1986) to estimate the object boundary in the image. This edge detection algorithm applies Gaussian smoothing and computes the intensity gradients to detect a wide range of edges in images. We can choose a threshold to filter out the edges that would be useful for the object of our interest. An example of Canny Edge detection is shown in Figure 3 where we could detect the edges of the objects present in the image.

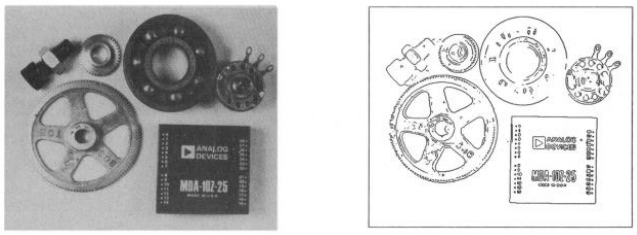


Figure 3: Canny Edge Detection Example (Canny, 1986)

By using this method, we can compute the reference point coordinates within a fraction of a second. Although it requires a specific setup while recording the video, the results from this method are accurate and close to the actual object boundaries. So, we preferred this method to compute the coordinates of the reference point.

3.2. Critical Surface Identification

Critical surface areas on the mandrels refer to the sealing surfaces (point of contact) for the O-rings. Any damage and defects in these areas result in oil spilling and hence the mandrels should be reused, repaired, or discarded based upon the degree of the damage. To identify and project the critical surfaces (O-ring grooves) in the frames of the video, the previously mentioned fixed setup is used where the camera remains stationary. The concepts of camera projection (Collins, 2007) are then used to convert the real-life coordinates of the grooves to pixel coordinates in the image. The projection equation shown in Eq. (1) is used to achieve this.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \underbrace{\begin{bmatrix} -f/s_x & 0 & +o_x & 0 \\ 0 & -f/s_y & +o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{M_{int}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{M_{ext}} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \quad (1)$$

Here, U, V, and W are the real-world coordinates that need to be converted into the pixel coordinates (u, v) of the image. Variables s_x and s_y are the pixel size of the camera sensor along the x and y axes respectively, further explained in Eq. (13) and Eq. (14). The focal length of the camera is denoted by f and the camera's film plane center offsets from its sensor's pixel array origin along the x and y axes as explained in Eq. (9) and Figure 8 are denoted by o_x and o_y.

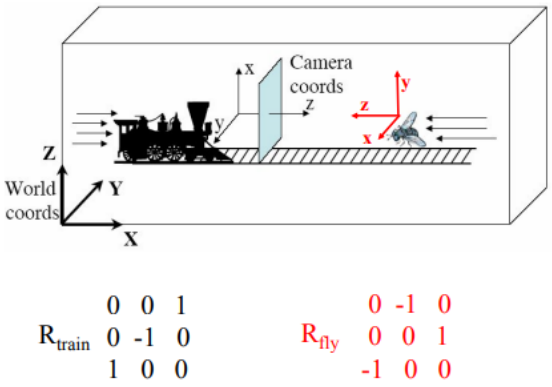
3.2.1. Extrinsic Parameters

The real-world coordinates are first converted into camera coordinates using extrinsic parameters R in Eq. (2) and T in Eq. (3) which together form M_{ext} in Eq. (1) and enable this transformation.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (2)$$

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad (3)$$

R is the measure of rotation required to align the real-world and camera coordinate axes. This means that (r_{11}, r_{12}, r_{13}) is rotational units needed in the camera x-axis, (r_{21}, r_{22}, r_{23}) in the camera y-axis, and (r_{31}, r_{32}, r_{33}) in the camera z-axis. This can be better understood by looking at Figure 4. As you can see in this figure, the train camera's x-axis resonates with the real-world z-axis, therefore (r_{11}, r_{12}, r_{13}) becomes (0, 0, 1). The y camera axis resonates with the real-world x-axis but is in opposite directions, hence (r_{21}, r_{22}, r_{23}) becomes (0, -1, 0). The Z camera axis resonates with the real-world x-axis, therefore (r_{31}, r_{32}, r_{33}) becomes (1, 0, 0). This way we get R_{train} mentioned in Figure 4.



$$R_{train} = \begin{matrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{matrix} \qquad R_{fly} = \begin{matrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{matrix}$$

Figure 4. Extrinsic Parameter – Rotation (Collins, 2007)

T, on the other hand, is the location of the camera relative to the real-world coordinate system. t_x, t_y, t_z are, therefore, the camera's position on world coordinates' x, y, and z-axis respectively. They are a measure of translation/movement that the camera axis needs to undergo to be aligned to the real-world axis as shown in Figure 5.

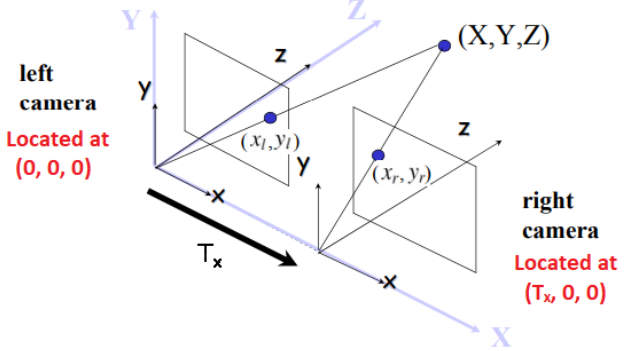


Figure 5. Extrinsic Parameter - Translation (Collins, 2007)

3.2.2. Perspective Matrix Equation

The concept of perspective projection helps us to understand how 3D coordinates can be projected to a 2D plane. This is then employed to convert the 3D camera coordinates which we calculate using extrinsic parameters, to 2D film coordinates. The film plane of the camera is located at f (focal length) units along with the optic (Z) axis of the camera coordinate system (see Figure 6).

The perspective projection equations i.e., Eq. (4) and Eq. (5) can be derived from the rule of the similar triangles (see Figure 6 and Figure 7)

$$x = \frac{fX}{Z} \quad (4)$$

$$y = \frac{fY}{Z} \quad (5)$$

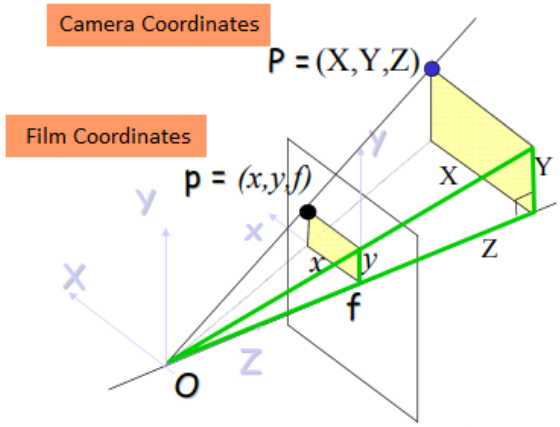


Figure 6. Projection on Film Plane (Collins, 2007)

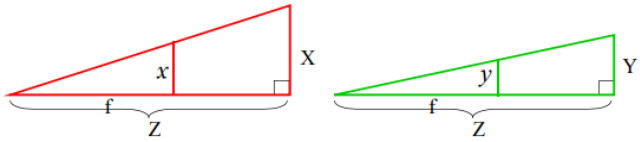


Figure 7. Object and projection on film plane forming similar triangles (Collins, 2007)

The perspective projection equations can also be represented as a matrix by introducing homogenous coordinates. Homogenous coordinates represent a 2D point (x, y) by a 3D point (x', y', z'), by adding a fictitious third coordinate. Given (x', y', z'), the 2D point can be recovered as shown in Eq. (6) and Eq (7).

$$x = \frac{x'}{z'} \quad (6)$$

$$y = \frac{y'}{z'} \quad (7)$$

This way, we transform the perspective projection equation (Eq. 4 and Eq. 5) into a matrix (Eq. 8).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (8)$$

3.2.3. Intrinsic Parameters

The images projected on the film plane are further digitized, which poses the need for us to transform the film coordinates of interest derived from the perspective projection matrix into pixel arrays.

This is achieved by the usage of intrinsic parameters O and S and shown in Eq. (9) and Eq. (10) respectively.

$$O = [O_x, O_y] \quad (9)$$

$$S = [S_x, S_y] \quad (10)$$

O is the offset or the image center in the film plane that needs to be added to the derived film coordinates, as the film and pixel coordinate systems along with their origins are different (seen Figure 8).

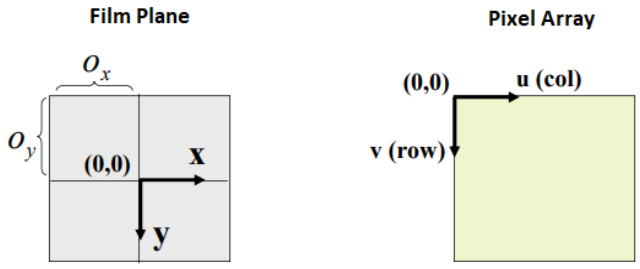


Figure 8. Offsets O_x and O_y (Collins, 2007)

With this, we can calculate the pixel coordinates (u, v) as per Eq. (11) and Eq. (12).

$$u = x + o_x \quad (11)$$

$$v = y + o_y \quad (12)$$

Conversely, we can calculate o_x and o_y if we find the pixel and film coordinates for any one point.

The second intrinsic parameter i.e., S represents the pixel size s_x and s_y. s_x and s_y give us a measure of how many units of distance (mm, inch, etc.) a pixel covers on the image plane (camera sensor). It can easily be derived as in Eq. (13) and Eq. (14).

$$s_x = \frac{\text{sensor width}}{\text{image width in pixels}} \quad (13)$$

$$s_y = \frac{\text{sensor height}}{\text{image height in pixels}} \quad (14)$$

The pixel size is incorporated into Eq. (11) and Eq. (12) effectively changing them to Eq. (15) and Eq. (16)

$$u = \frac{x}{s_x} + o_x \quad (15)$$

$$v = \frac{y}{s_y} + o_y \quad (16)$$

3.2.4. Projecting Critical Surface Grooves

Since the mandrel being cylindrical, is a circle in the real-world Y-Z plane, we can get the real-world Y and Z coordinates of the grooves (red bands in Figure 9) along the circular surface with the help of the mandrel's radius (as shown in Figure 9).

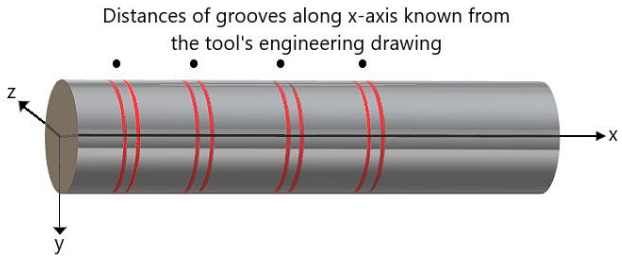


Figure 9. Figure showing a Mandrel and the critical surface grooves

The X coordinates of the grooves are taken from the engineering design of the tool. These real-world (U, V, W) coordinates are used along with the extrinsic and intrinsic parameters as shown in Figure 10 to get the pixel coordinates (u, v) of these grooves in the images.

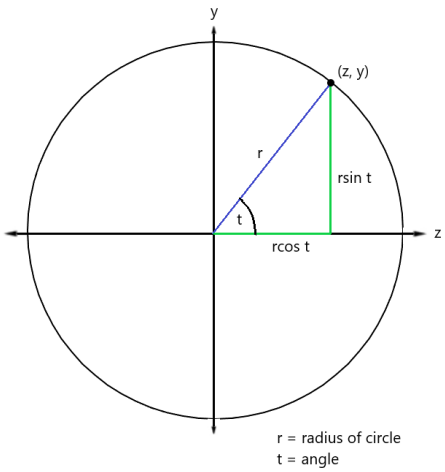


Figure 10. Offsets O_x and O_y

3.3. Damage Detection

To detect the damage on the mandrel in the image, we use a computer vision technique called Semantic Segmentation in conjunction with Deep Learning. We train a model based on U-Net architecture to predict which pixels in each image were damaged.

3.3.1. Semantic segmentation

Semantic segmentation is a process where every pixel in an image is associated with a certain label or class. It helps us to distinguish between predefined multiple categories in an image with pixel-level granularity (as in Figure 11).

Figure 11. Semantic Segmentation predicting 2 classes (Dog and Background)

3.3.2. U-Net

We use U-Net architecture (Ronneberger, Fischer and Brox, 2015) to employ Deep Learning and achieve Semantic Segmentation. U-Net has 2 logical components, an encoder, and a decoder. The encoder, which can be any Convolutional Neural Network (CNN) architecture-based feature extractor, extracts feature maps from images at a resolution and downsamples the images before repeating the same process. This way we get feature maps of an image at different resolutions. The decoder takes the feature maps of the lowest resolution and upsamples them. It then takes the upsampled feature maps and merges them with the feature maps that were extracted at that resolution earlier by the encoder. This process is repeated by the decoder till we get the final output with the original resolution of the input images. This output is the representation of the various categories present in the images and the pixels belonging to them.

4. DATA AND EXPERIMENTS

4.1. Reference Point Identification and Critical Surface Detection

We have created a fixed setup to rotate the mandrel and capture its surface area. The specifications of the camera are mentioned in Table 1. To capture all the surface area, the mandrel would be rotated along its axis at a fixed speed, so it would not distort the mandrel recording in the video.

Table 1. Camera Specifications

Specification	Value
Camera Used	Nikon D550
Sensor Size	23.5 mm x 15.60 mm
Pixel Size	0.0039 mm (For 24 MP Image)
Focal Length	18 mm

As shown in Figure 12, we fixed the camera on a stationary point within a certain distance to capture the surface area but not too far which would make the damage on the surface invisible to the naked eye. As mentioned before, the reference point can be easily detected from the mandrel boundaries in the image given the clear background. Based on our observations, The Camera Projection method can locate the critical surface area with a **maximum error of 2.54 mm** (0.1 inches) as per our experiments with our setup. If we expand the critical surface areas by 2.54 mm on either side, we can include all the actual critical surface areas in the image that would otherwise be missed due to this possible error. The final projections can be seen in the below figure (Figure 13) where the area enclosed between the white curves on the mandrel surface is our critical surface. The small black markings on the surface represent the markings of the actual critical surface. As we can see, our projections are closer to the actual areas and follow the conical nature of these curves along the surface.

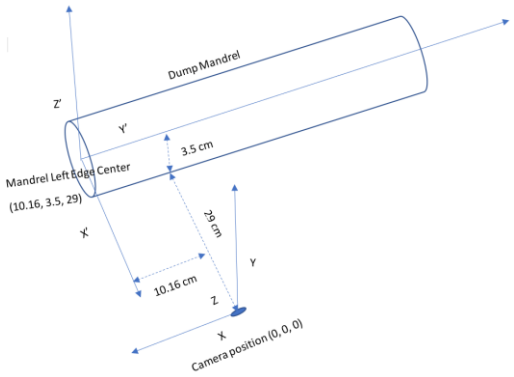


Figure 12. Camera Setup for Critical Surface Identification

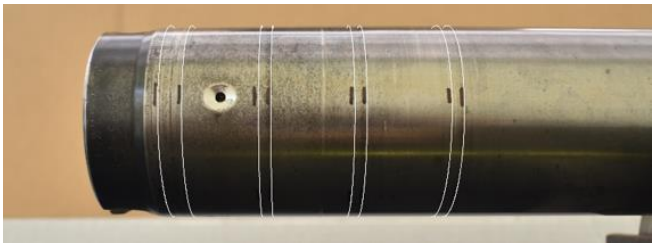


Figure 13. Critical Surface Projection

As we used a uniform background in our experiments (visible in Figure 13), the foreground (dump mandrel) stands out, and

the edge detection method could be used to compute the boundary of the mandrel in the image, which helps us to detect a reference point and project critical surfaces. If the background is noisy, the edge detection method alone won't be useful as the edges from any texture or object in the background can cause interference, and usage of deep learning based methods would be required to extract the boundary of the mandrel.

4.2. Damage Identification

In our first model training, we recorded 20 videos of the damaged mandrels for the damage detection model training. We converted these videos into frames and annotated them where the damage is visible with our naked eye. For semantic segmentation, we need to annotate on pixel level and assign them a class. For implementation purposes, we have combined all the damages that can occur on the mandrel into one class. So, we will be predicting two classes i.e., damage and background (no damage), which is a binary classification at pixel level and outputs the probability of each image pixel belonging to the two mentioned classes. A segmentation mask would be prepared to mark damage in the image, which is a 2D array of 1s and 0s where 1 indicates damage on the (x, y) coordinate in the image and 0 indicates background. The frames and their corresponding segmentation masks are used as input and output for training the semantic segmentation model.

The model takes a 4D array as input i.e., n images with RGB (Red, Green, and Blue) values, and returns a 4D array as output. The array returned as output would have, for every input image, the probability of the pixel being damaged as well as the probability of it being the background. Comparing these probabilities for every pixel of all the images, we get a mask (2D array) for every input image that tells us the image coordinates of damages and background (no damage). We used the previously discussed U-net architecture by Ronneberger et al. (2015) with ResNet (He, Zhang, Ren and Sun, 2016) and EfficientNet (Tan & Le, 2020) backbones. The EfficientNet-based U-net model fared better than the ResNet-based one for damage detection.

The model was trained by using EfficientNet architecture (Tan & Le, 2020) as the encoder for the U-net model (Ronneberger et al., 2015). We also employed transfer learning as the weights from a model pre-trained on ImageNet (Deng, Dong, Socher, Li, Li and Fei-Fei, 2009) were used. The dataset, which consisted of 300 images extracted from 20 videos of the tools rotating on their axis was split into train, test, and validation sets in a ratio of 70:15:15. The model was then trained by defining the decoder which could upsample features as per the U-net architecture, using Adam (Kingma & Ba, 2015) as the optimizer algorithm and Binary Crossentropy as our loss function.

Also, one whole image of the tool provided to the model as input would be compressed way too much horizontally (see Figure 14). This would, in turn, distort the dimensions of the damages as well and affect the model's capability to detect them. To overcome this, we decided to split any given input image into multiple slices (Figure 15) and send all of them as a batch for damage identification which can later be stitched back together after the model processes them into one whole output image.

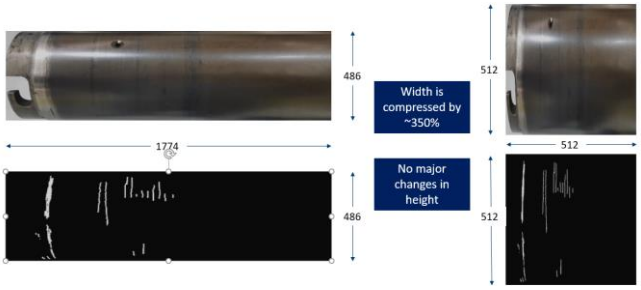


Figure 14. Original image and its damage representation (left) being compressed as the model input (right)

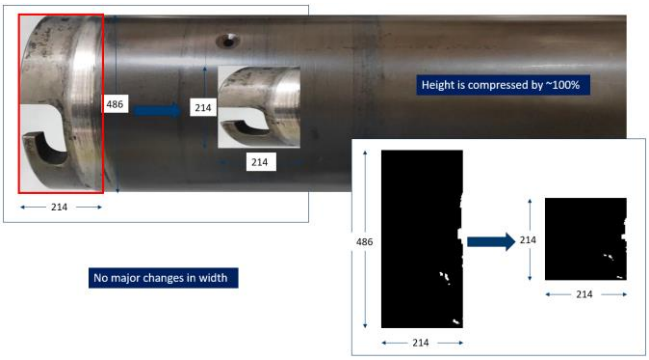


Figure 15. A slice of the input image (enclosed in red) and its damage representation before and after compression.

In general, average pixel accuracy across all classes is used as an evaluation metric for Semantic Segmentation models. As the occurrence of damage in the images is very low, the pixel accuracy metric did not help differentiate between good and bad models. So, we chose Dice Coefficient as our evaluation metric for the model. As mentioned in Eq. (17), the dice coefficient for input sample A and output sample B is defined as twice the intersection of the pixel area divided by the sum of the individual areas of the samples.

$$\text{Dice coefficient} = \frac{2(A \cap B)}{|A| + |B|} \tag{17}$$

The dice coefficient describes the percentage of overlap we can expect from the predicted pixels compared to ground truth pixels. With the above training, we observed a **mean dice coefficient of 0.7** which is good for locating the damage in the images. This means that SMEs can expect at least 70% of the damage to be detected in the critical surface areas. As the damage areas are very scarce and small in the tools and

training data, we expect we can increase the damage detection percentage with more data to train the model. As the model outputs a probability of damage on each pixel, we can apply a probability threshold to select the damages on the image. Although it is not clear in our use case, we noticed that the mean dice coefficient seems to be slowly decreasing and drops to a lower value when plotted against the probability thresholds. The huge drop in the mean dice coefficient values occurred for threshold values greater than 0.5. So, we chose a threshold where the mean dice coefficient is steady before dropping to a lower value, which provides good results when compared to the lower threshold values.

During experimentation, it was also noticed how lighting conditions could prove to impact damage detection adversely. As you can see in Figure 16, given the lighting conditions and the reflective surface of the tool, the features of the surface are hard to make out and any damage in that area would be unidentifiable.

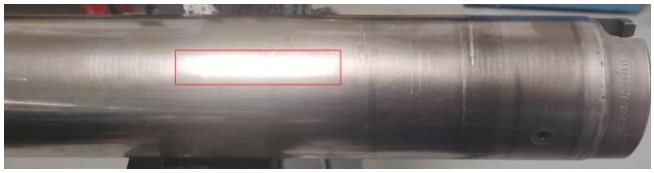


Figure 16. Reflections on tool due to unfavorable lighting conditions

4.3. Final Results

As our area of interest is the critical surfaces on the mandrel, we need the damage that was in these areas. To obtain that, we would require two image masks. First is a 2D image mask which contains a mask of critical surface areas on the mandrel. The second one is the output mask obtained from the above damage detection model. Then, we combine the above two masks by applying bitwise AND on the two image masks. This application filters out the pixels which have a value of 1 on both masks. The result can be seen in the below figure (Figure 17) where the red pixels show the damage on the surface and the white curves enclose the critical surface area on the mandrel.

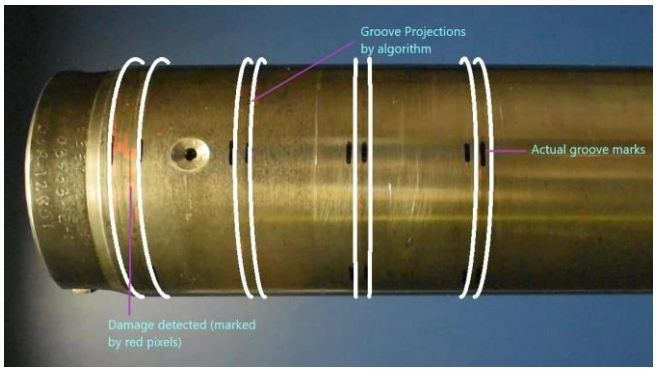


Figure 17. Damage within Critical Surface Area

The software for the experiments mentioned in this paper is developed in Python using OpenCV (Bradski, 2000) library for image processing. We have used an NVIDIA Tesla K80 GPU with 12 GB RAM and a CPU of 16 GB RAM for training and inference of the U-net model. Using this hardware, we were able to compute the inference of an image in 0.2s and computed the final solution where damage is shown in each frame, with 5 FPS. If we increase the computing power with an efficient GPU with more RAM, we can increase the inference speed to 20-30 FPS which would make the solution a real-time one.

To decide whether the tool can be used for the next job, we compute the percentage of red pixels on the critical surface area. We average these percentage values over all frames and compute the overall damage percentage of the mandrel. We can then keep a threshold to identify which tools would require maintenance, which ones would be fit for the next job, and which would require a manual inspection when the damage percentage is neither too high nor too low.

For keeping a suitable threshold, we can analyze the defective mandrel videos and come up with a threshold that would be ideal for automatically deciding the reusage of the tool. So, by using the framework mentioned in the paper, we can make the inspection process, from identifying critical surface areas to damage detection to the final decision of tool reusability, completely automatic.

5. CONCLUSION

The framework provided in this paper can be used to digitally detect surface damage for any kind of tool whose geometry can be mathematically modeled. With the fixed setup, we only need the engineering designs of a tool to select the critical area and detect the damage on that area. Further, if we have gathered more data, we can also train the damage detection model for all kinds of available damages for a better decision-making process, as one kind of damage that occurs with less frequency can hamper the usability of the tool more than another kind of damage occurring in higher frequency. Also, with more data, we can increase the accuracy of the damage detection model and the solution gets better with each input. We are hopeful that this framework can be used to reduce the time in manual SME inspection and help them perform the task in real-time.

NOMENCLATURE

SME Subject Matter Expert
PASCAL Pattern Analysis, Statistical Modelling, and Computational Learning
3D 3 Dimensional
2D 2 Dimensional
IoU Intersection over Union
4D 4 Dimensional
RGB Red Green Blue
CNN Convolutional Neural Network

GB Gigabyte
RAM Random-access memory
GPU Graphics Processing Unit
FPS Frames Per Second

REFERENCES

Berveglieri, A., & Tommaselli, A. (2018). Reconstruction of Cylindrical Surfaces Using Digital Image Correlation. Sensors 18, no. 12: 4183. https://doi.org/10.3390/s18124183.

Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

Canny, J.F. (1986). A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8, 679-698.

Collins, R. (2007). Camera Projection (Extrinsics). CSE/EE486 Computer Vision I. Fall 2007. Penn State University. Lecture.

Collins, R. (2007). Camera Projection (Intrinsics). CSE/EE486 Computer Vision I. Fall 2007. Penn State University. Lecture.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database, Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (248-255), June 20-25, Miami, FL, USA. doi: 10.1109/CVPR.2009.5206848.

Gruen, A. W. (1985). Adaptive Least Squares Correlation: A Powerful Image Matching Technique. South African Journal of Photogrammetry, Remote Sensing and Cartography, vol. 14, pp. 175–187.

Gruen, A. (1996). Least square matching: A fundamental measurement algorithm. In Close Range Photogrammetry and Machine Vision. Bristol, UK: Whittle Publishing, pp. 217–255.

He, K., Zhang, X., Ren, S., & Sun, J., (2016). Deep residual learning for image recognition, Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), (770-778), June 27-30, Las Vegas, NV, USA. doi: 10.1109/CVPR.2016.90.

Kingma, D. P., Ba, J., (2015). Adam: a method for stochastic optimization, Proceedings of International Conference on Learning Representations, May 7-9, San Diego, CA, USA.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. Proceedings of Medical Image Computing and Computer-Assisted Intervention (MICCAI), (234-241), October 5-9, Munich, Germany. doi: 10.1007/978-3-319-24574-4_28.

Tan, M., & Le, Q. (2020) EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, Proceedings of the 36th International Conference on Machine Learning, PMLR 97:6105-6114.

BIOGRAPHIES

Gautam Kumar Vadisala is a Lead Data Scientist at Schlumberger Technology Center in Pune, India. He has his bachelor's degree (B.Tech.) in Electronics and Electrical Engineering from the Indian Institute of Technology Guwahati. His main interests are Computer Vision, Deep Learning, Machine Learning, and Data analytics.

Anurag Singh Rawat is a Machine Learning Engineer at Schlumberger Technology Center in Pune, India. He has his bachelor's degree (B.E) in Computer Science from Rajiv Gandhi Technical University. His area of focus is Machine Learning based scalable solutions on the cloud.

Abhishek Dubey leads the AI Product Development at Schlumberger Technology Center in Pune, India. He has an MS degree in Computer Science from the Indian Institute of Science, Bangalore. His area of expertise is building large-scale, end-to-end Machine Learning Products on the cloud. His main area of research is Deep Learning for structured & unstructured data and prognostics & health management.

Gareth Yen Ket Chin is a Senior Mechanical Engineer at Schlumberger. He has his bachelor's degree (B.E.) in Mechanical Engineering from the National University of Singapore. His main interests are Industry 4.0 Revolution, Data-driven Prognostic and Health Management (PHM) systems with Machine Learning (ML), and Deep Learning (DL) techniques.

Fabio Abreu is a Manufacturing Support Manager at Schlumberger. He has a master's degree in Reliability Engineering from the University of Tennessee – Knoxville.

APPENDIX

Although we haven't used it in our final solution, we have worked on a method that processes the video and creates an image that displays the total surface area of the mandrel. This method unwraps the whole mandrel surface and displays it on an image that would be ideally impossible to visualize at one time.

To achieve that, we process the video into frames and locate the mandrel in each frame using the previously mentioned

Camera Projection method. Then we extract a small horizontal strip of about 10 pixels (depending upon the length of the video) around the mandrel central surface in each frame and append them to create an image that displays the overall surface area. We use some reference points such as holes, grooves, etc. to mark a complete rotation of the mandrel in the video and not to over-represent the mandrel area on the unwrapped image. For example, if we assume the holes on the mandrel surface as reference points for rotation comparison, we note down the location of these holes in the first frame and stop the solution when we encounter the same holes again within some pixel distance of the original location.

For better damage detection, we can horizontally segment the unwrapped image into segments of 5 or 10 and combine the individual segment outputs for final damage detection of the unwrapped image. The final output can be seen in Figure A-18 where white vertical lines enclose the critical surface area. The red pixelated area within these lines shows the damage on the surface. One thing to note is that since we took a small strip around the mandrel center, our critical surface boundaries would not have a conical shape. Also, we can observe that some holes on the mandrel surface appear expanded, and others appear shrunk. This occurs because of the change in rotation speed within different points of the mandrel rotation. When the rotation speed decreases relatively, we will get more surface area in the unwrapped image and vice versa.

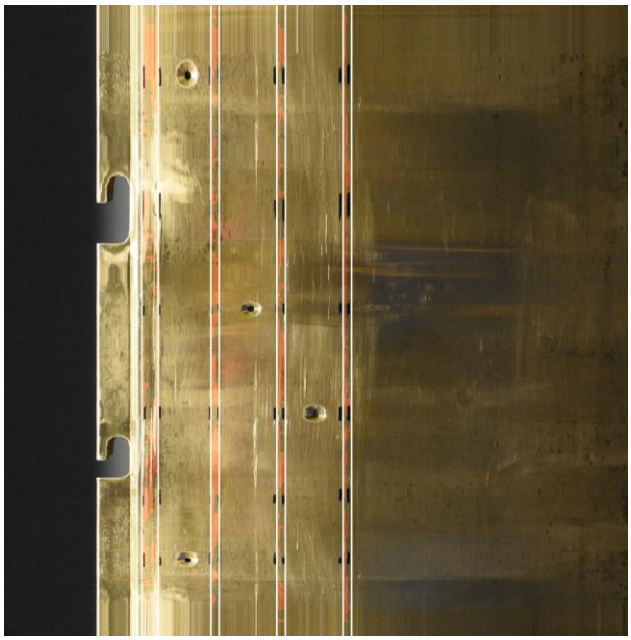


Figure A-18. Damage on the Unwrapped Mandrel

Here too, we can take a percentage of red pixels within the critical surface and create a threshold to come up with a decision on using the mandrel for the next job or not. If the mandrel doesn't maintain a fixed speed, we will not obtain an even spaced image of all its contents. Because of this reason, we did not select this method to come up with the final decision of reusing the mandrel as we would give more preference to the damage where the mandrel was recorded relatively slower than the other parts. But if the rotation is fixed, we can see the even spaced points across the unwrapped image. Nevertheless, this method can be useful to extract the surface area of the cylindrical objects in videos and further detect the damage on the surface.