**Segmentation, characterization and superimposition of PSP damage-induced dental X-ray artifacts**

Wenda Zhao

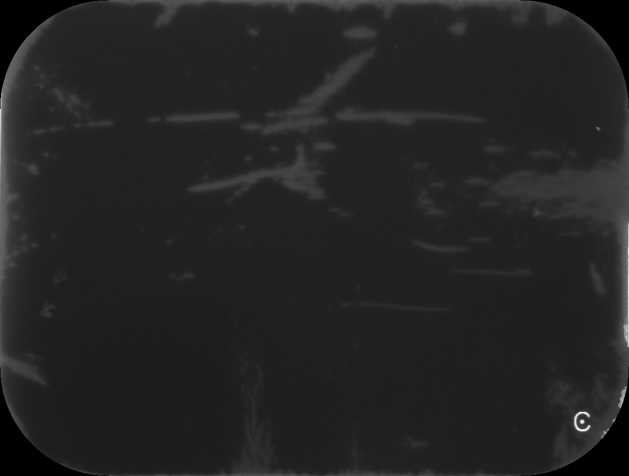
Contributors: Hershel Stark, John Valen, Trevor Thang

Pascal Tyrrell Lab  
Department of Medical Imaging  
University of Toronto

**Introduction**

Phosphor storage plates (PSPs) are commonly used sensors for digital intraoral radiography. They are made of a polyester base coated with phosphor particles which store a latent image in the form of energy when excited by X-ray. The energy is then released as blue fluorescent light under a helium-neon laser beam and the image is shown on a computer system1. In comparison to this indirect way of acquiring images, complementary metal oxide semiconductor (CMOS) sensors are directly connected through a cable to the computer and can display the image immediately after exposure. PSPs have the advantage of allowing easier placement in the oral cavity and causing less discomfort to the patients2. These sensors are meant to be reusable. However, after a prolonged period of use, damages accumulate on the plates and can introduce artifacts to the acquired images (see figure 1). CMOS sensors, on the other hand, are free of such damages. A study in Turkey showed that 20% of the images obtained via PSPs over a six-month period contained visible artifacts3. And PSPs-induced artifacts have been shown to adversely affect the diagnostic ability in digital mammography4. Thus, this has raised the concerns of possible missed diagnoses when using damaged sensors in dental radiography. Clinics are now dealing with this problem by discarding the plates after a certain times of use. Previous study comparing images of PSPs with different used times has suggested that each plate can be used up to 200 times without showing statistically significant changes5. However, in real life this decision is made rather arbitrarily by clinicians. Thus, a standard model needs be established to characterize and quantify the artifacts and to give an accurate prediction as in whether a damaged sensor would impede diagnosis or not based on its characteristics. Clinicians can therefore be informed of when to discard a plate without hurting diagnosis, as a consequence of doing so too late, or wasting money, as a consequence of doing so too early. In order to accomplish this, two preliminary goals need to be achieved in preparation for building such model. The first is the characterization of dental image artifacts, which the model would then take as the input. Previous attempts have been made to categorize these artifacts. In a study done by Kalathingal *et al.*, the authors subjectively rated the damaged plates into five categories of severity in order to study the effect of wiping as a means to remove surface contaminations6. And in a recent paper published by Çalışkan and Sumer, the artifacts were placed into five categories based their appearances: cracking, scratches, peelings of edges, bite-marks and crescent-shaped bending7. These attempts have opened up discussions on this topic but the findings were also at high risk of subjectivity and bias. Thus, a more robust approach to segment and characterize these artifacts is needed. The second goal is that we need to establish a “gold standard” of whether a provided artifact will affect diagnosis. And in order to achieve that, dental images with pathologies needs to be provided in the background of a representative sample of artifacts for dentists to examine. However, acquisition of such images in clinics is unlikely since it is considered unethical to knowingly use damaged sensors on patients. One way solve this problem is to superimpose images of the plates along onto artifact-free teeth images taken with CMOS sensors, and use the superimposed images for testing.

Thus, in preparation for building a model that can predict the effect of dental image artifacts on diagnosis, we face the question of whether we can segment, characterize those artifacts and superimpose them onto clean teeth images. And we hypothesize that we can achieve that by using an algorithmic and machine learning approach. Our objectives are to segment the artifacts from the image; to quantify and characterize them; to cluster the segmented artifacts into meaningful groups based on the characterization; and to superimpose images of PSPs containing artifacts onto clean teeth images taken with CMOS sensors to mimic real dental images taken on damaged PSPs.



**Figure 1.** **Phosphor Storage Plates with damage-associated artifacts.** Images from artifact image 29 (left) and 60 (right).

**Materials and Methods**

**Fast normalized cross-correlation**

We obtained 339 PSPs with various degrees of damage from the clinics of the Faculty of Dentistry at University of Toronto.

There is a “C” shaped object at the right bottom corner of all the PSP images (see figure 1). It is marked on the sensor itself to help with orienting the plate. But for our study, this “C” shape could potentially be picked up by the segmentation algorithm and interfere with the process. Thus, we utilized the scikit-image8 package in python for template matching to locate and remove the “C”. The algorithm uses fast normalized cross-correlation9 to find the occurrence of a temple in an image. We cropped out a 40 pixels by 40 pixels piece of one original PSP image with the “C” in the middle to serve as a template for the algorithm. The algorithm returns a correlation matrix showing the similarity between the template and pixels across the entire tested image, with the maximum being the centre of the matched area. We then set all pixels within a radius of 20 from the matched point whose value is larger than 45 to a value of 20.

**Density-based spatial clustering of applications with noise (*DBSACN*)**

*DBSCAN* is a data clustering algorithm initially proposed by Ester *et al.*10 in 1996, and does, as the name suggests, density-based clustering that groups spatially close points together. The algorithm searches the *ɛ* neighborhood of a point, which is the multidimensional space within a distance *ɛ*, calculated by some distance metrics, usually Euclidian distance, from that point. If there are more than a certain number of points, denoted *MinP*, in that neighborhood, the data point is then considered a core point and a cluster is started. All the points within the neighborhood of a core point are termed reachable from the core point and are all part of that cluster. The *ɛ* neighborhoods of all cluster members, if again contains more than *MinP* points, are also termed reachable from the original core point and are thus grouped into the same cluster, so on and so forth. In other words, reachability can be passed on from one core point to another core point in its neighborhood. Any point that is not reachable from other points is grouped into noise. The *DBSCAN* algorithm contains two hyperparameters: *ɛ* and *MinP*. In our study, we chose 20 for both based on clustering results from several testing values. In order to reduce the noise, we also thresholded our images by setting any pixels below 65 to 0. We used the scikit-learn11 package in python for *DBSCAN* implementation. Our inputs are in the form of sparse matrices: SciPy12 COO matrices of triplet (row, column, intensity) tuples. The algorithm returns a cluster label for each pixel on the image.

**Feature computation**

We picked the following 9 features to be calculated from each segmented artifacts: number of pixels to represent the area of artifacts; sum of intensity to represent the volume; range along *x* and *y* axes to represent the length and width; *x* and *y* coordinates of the centroid to represent the location; maximum and median intensity to represent the intensity; and a categorical variable of shape to be either line, triangle, rectangle or circle. Because centrosymmetric locations on the plates are equivalent in clinical settings, we used the location of centroid with respect to the centre of the image instead of to the origin of the coordinate plane, which is usually the bottom left corner. We also omitted minimum intensity as a feature because we applied a lower threshold of 65 during *DBSCAN* segmentation, which made 65 to be the minimum in many artifacts.

For shape detection, we utilized the contour of the segmented artifacts. An OpenCV13 package for contour approximation was used to approximate the contour with a shape of fewer vertices within a range of error *ɛ*, which we picked to be 4% of the contour perimeter. The algorithm is an implementation of the Ramer–Douglas–Peucker algorithm14, 15, and it returns a list of vertices for the approximating shape. With two vertices, it means the shape of interest can be approximated by a line; whereas three vertices indicates a triangle; four being a rectangle. We put any artifact that can be approximated by a polygon with more than four vertices into the category of circle.

***K-prototypes***

*K-prototypes* is a clustering algorithm that can be used to partition large data sets containing both numerical and categorical variables into *k* homogeneous groups, where k is a hyperparameter representing the number of clusters desired16. The algorithm is based on the *k-means* paradigm that minimizes a dissimilarity metrics between data points and clusters through multiple iterations. In *k-means*17, which can only be used for numerical variables, the dissimilarity is measured by a distance function, usually of Euclidean distance, between the data points and the centroid of the cluster. The centroids are first randomly initiated. After computing the distance metrics, every data point is assigned to its closest cluster. Then the centroids of clusters are recalculated based on the updated results and the next iteration of computing distance and reassignment starts. This will go on until convergence where assignments no longer change. Another derivation of *k-means* called *k-modes*18 is used for categorical variables only. It uses the number of mismatches between the data points and the modes of the clusters as a measure of dissimilarity and also goes through multiple iterations of computing and reassigning until the algorithm converges. *K-prototypes* combines the two algorithms together and calculates the dissimilarity by a weighted combination of that of *k-means*, for numerical variables, and that of *k-modes*, for categorical variables, and is thus able to perform clustering on data sets containing both types of data. Here we used a package developed by Nico de Vos for python implementation of *k-prototypes*. All numerical variables were standardized before applying *k-prototypes* by using a MinMaxScaler from the scikit-learn package that scales each feature to a given range:

where [*min*, *max*] is the range the data is to be scaled to, and in our case, [0, 1].

**Contrast limited adaptive histogram equalization (CLAHE)**

Histogram equalization is a way to improve the contrast of images by spreading out the frequently appearing intensity values. In our case, this could prove to be helpful because it can potentially show the subtle artifacts present on the plates that are not necessarily visible to human eyes. Global histogram equalization works by transforming the entire image to obtain a linear cumulative distribution function of the histogram20. However, it also comes with the problem of over-brightness and noise. Instead, CLAHE performs histogram equalization on small blocks of the image and applies contrast limiting, which clips any histogram bin above a certain contrast limit before equalizing, to reduce noise21. Block size and clip limit are two hyperparameters that can be adjusted. We used an OpenCV implementation of CLAHE with a block size of 64 pixels by 64 pixels and a clip limit of 4.0 based on the performance of various testing values.

**Superimposition**

Our superimposition was experimented on 25 selected artifact images from the 339-image-bank. All artifact images were superimposed onto one CMOS image of teeth taken from a dental mannequin. Images taken directly with the same 25 damaged plates on the same mannequin were also acquired for validation purposes.

Several algorithms were tested for superimposition of image artifacts onto clean dental images. The simplest way is to linearly combine the two images with different weights for each:

Here *a* is a matrix of the artifact image in the form of numpy array22, *b* is the clean teeth background and *c* is the final superimposed image. *W1* and *w2* stands for the constant weight applied to the artifact and background.

Given the complexity and heterogeneity of the artifacts within one image, one might consider applying different weights for different intensity regions of the artifact masks:

Here *a*(*k*) corresponds to the *k*th intensity region of the artifact image containing all elements of *a* with intensity values between *mk* and *Mk*. To incorporate all possible intensity values of our 8-bit grey scale images, *mk* of the first region and *Mk* of the last region should be 0 and 255, respectively. And the upper limit of one region should be the same as the lower limit of the next region. *W1k* is the weight designated for the *k*th region of the artifact masks and *w2* is a constant weight applied to the entire image of the teeth background.

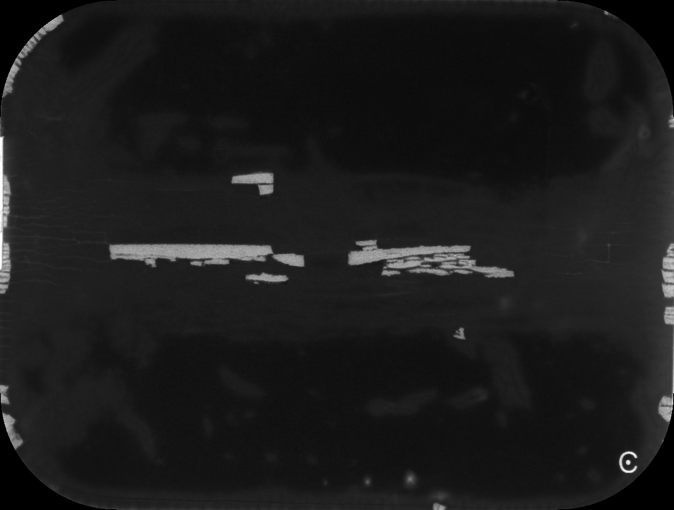
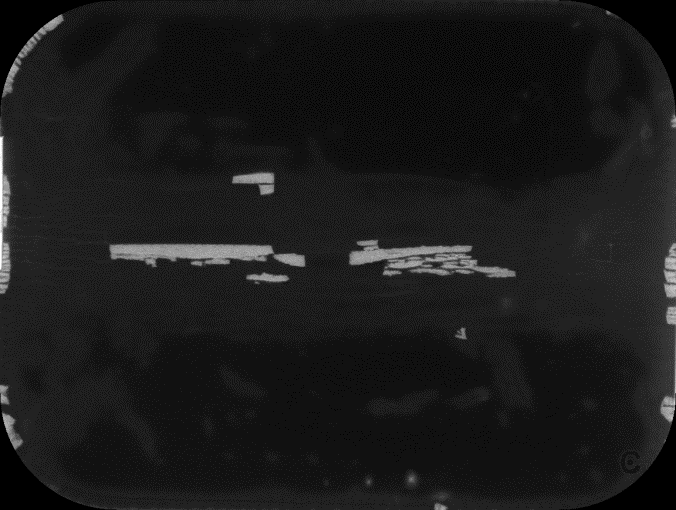
Instead a discontinuous weighting function such as in the thresholded weighting approach, our other attempt involves using a continuous weighting function that assigns weight for each pixel to be a function of their intensity. Based on the assumption and observation that subtle artifacts need to be more prominent in the final superimposed image than those readily bright artifacts, we chose a continuous, monotonically increasing function, exponential function, to be our weighting function:

Thus, the final image was constructed element-wised where the value of each superimposed pixel *cij* was the sum of the corresponding pixel *bij* in the teeth background weighted again by a constant *w2* and the corresponding pixel *aij* in the artifact masks weighted by *wij*, an exponential transformation of its own intensity. *A* and *B* are the constant coefficient and the base of the exponential function.

**Results**

**“C” removal**

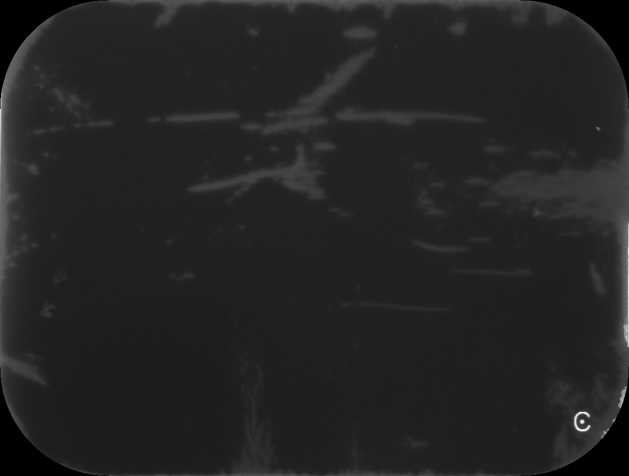
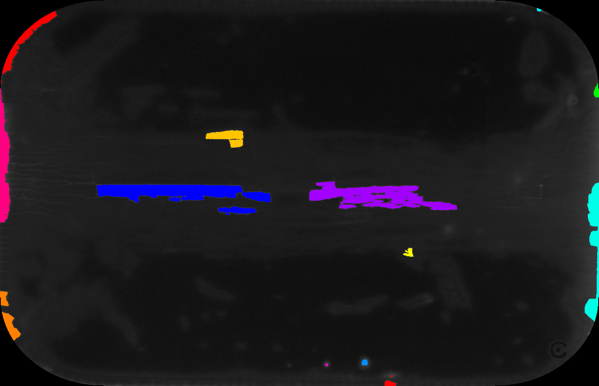
After implementing the template matching algorithm using a “C” template cropped out from one of the images, the “C”s in all artifact images were successfully located. It was then removed by setting all pixels within a radius of 20 from the matched point whose value is larger than 45 to a value of 20 (see figure 2).



**Figure 2.** **Removal of the “C” in artifact images.** Image from artifact image 60. The bright “C” at the right bottom corner of the artifact image was no longer present (see red arrow). However, a dark negative image of what used to be the “C” could still be seen.

**DBSCAN**

The DBSCAN algorithm was able to segment the artifacts with some success (see figure 3). Large, bright artifacts were easily picked up, whereas some of the more subtle, dimmer artifacts were not detected. Artifacts around the edges are generally better captured by the algorithm. Based on the results, we were able to create images with single segmented artifacts on a black background (see figure 4) to be used for further artifact-wise feature extraction.



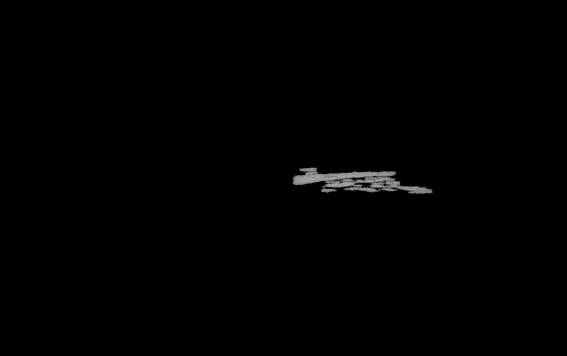
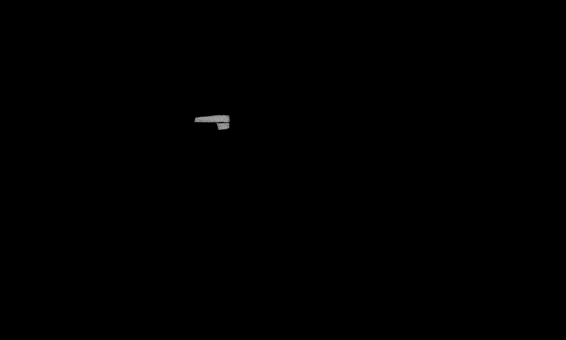
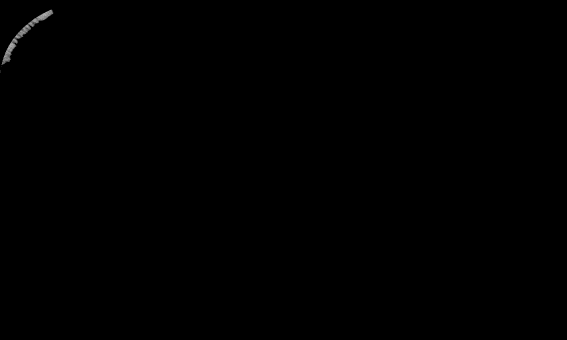
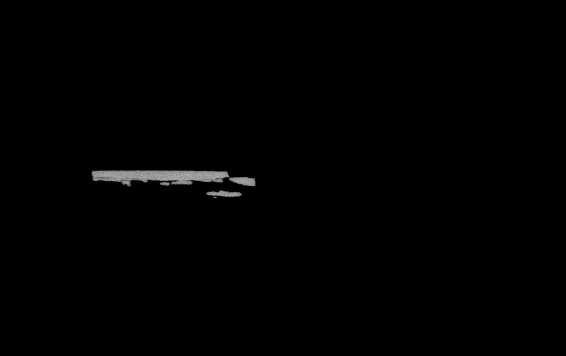
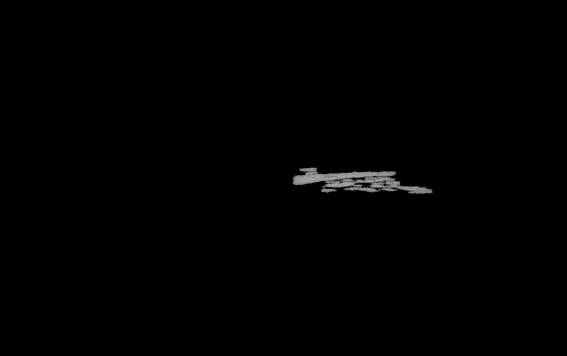
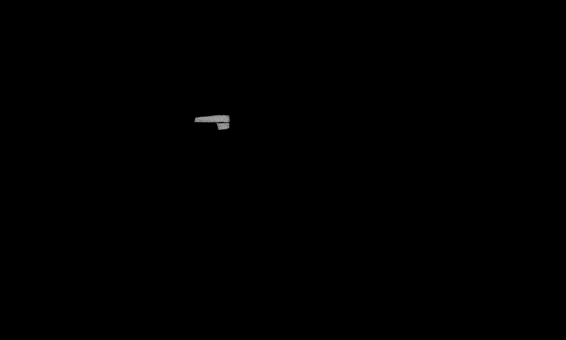
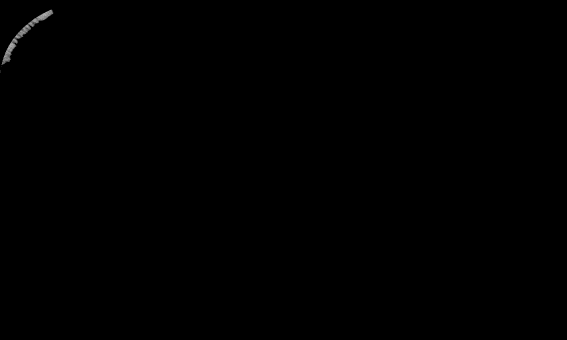
**Figure 3. Artifact detection by DBSCAN.** Image *a* and *b*: artifact image 60; image *c* and *d*: image 29. Parameters: *ɛ* = 20, *MinP* = 20. All images were thresholded at 65. Each detected artifact was represented in a unique color. Large, bright artifacts (see red arrows) were able to be detected, whereas many subtle, dimmer artifacts (see white arrows) were missed.

*a*

*b*

*c*

*d*



**Figure 4. Individual segmented artifacts.** Image *a*-*d* are different artifacts in artifact image 60 segmented by DBSCAN algorithm.

*a*

*b*

*c*

*d*

**Feature computation and *k-prototypes* clustering**

We calculated 9 features for each segmented artifacts, namely area, volume, length, width, centroid *x* location, centroid *y* location, maximum intensity, median intensity and shape. The first eight features are numerical variables whereas shape is a categorical variable of 0, 1, 2 and 3, representing line, triangle, rectangle and circle, respectively. We performed *k-prototypes* clustering on these 9 features with *k*=5 and obtained 5 clusters. To better visualize the results, we used boxes of 5 unique colors to show the artifacts (see figure 5). Out of the 1910 artifacts that we segmented from 339 images, 261 of them belong to Cluster 0, represented in blue; 279 artifacts are in Cluster 1, represented in red; 532 in Cluster 2, shown in green; 280 in Cluster 3, shown in pink and 558 in Cluster 4, shown in yellow. We calculated the mean of all features for each cluster (see table 1) and, based on the value and relative magnitude of those means, gave a qualitative description of the prototypical artifact for each cluster (see table 2).

**Figure 5. Clustering of artifacts by *k-prototypes*.** Examples here are from artifact image 29 (left) and 60 (right). Each color represents a different cluster determined by the algorithm: Cluster 0, 1, 2, 3, 4 shown in blue, red, green, pink and yellow, respectively (see arrows).

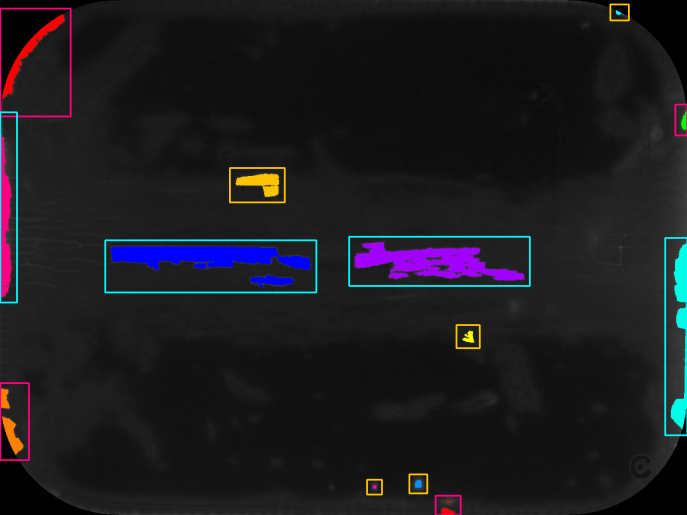
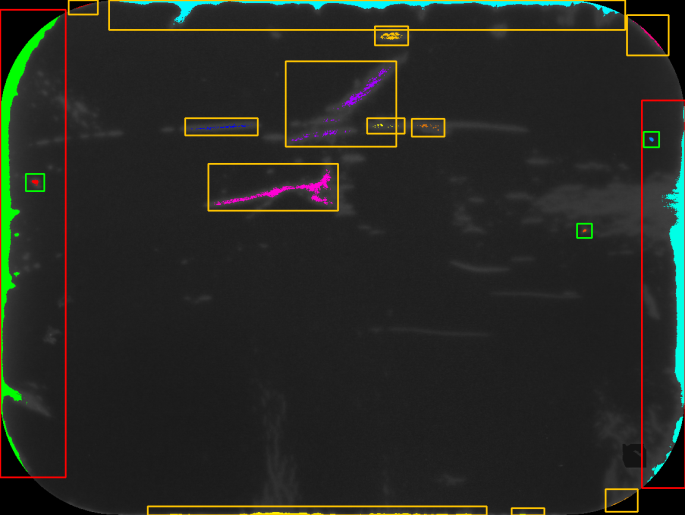


Table 1: Mean feature values of artifact clusters.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cluster | NumPixels | Volume | DeltaX | DeltaY | CentroidX | CentroidY | Max Intensity | Median Intensity | Shape |
| 0 | 998 | 150486 | 11 | 176 | 569 | 80 | 227 | 181 | 0.53 |
| 1 | 8487 | 679394 | 85 | 586 | 559 | 76 | 189 | 75 | 2.20 |
| 2 | 1278 | 101825 | 12 | 184 | 551 | 115 | 126 | 75 | 0.83 |
| 3 | 512 | 81779 | 65 | 33 | 436 | 364 | 195 | 151 | 1.51 |
| 4 | 245 | 19266 | 79 | 16 | 278 | 357 | 93 | 73 | 1.83 |

\*Means for the first eight variables were around to the nearest integer. Means for the categorical variable shape were rounded to two decimal points.

Table2: Descriptive features of artifact clusters

|  |  |  |  |
| --- | --- | --- | --- |
| Cluster | Color | # of artifacts in cluster | Description |
| 0 | Blue | 261 | Median sized narrow linear peelings of the lateral edges with high intensity |
| 1 | Red | 279 | Large narrow rectangular peelings of the lateral edges with median intensity |
| 2 | Green | 532 | Median sized triangular peelings of the lateral edges with median intensity |
| 3 | Pink | 280 | Small triangular peelings of the corners with high intensity |
| 4 | Yellow | 558 | Small rectangular blobs in the mid area of image with low intensity |