CS4302 - Signal Processing Coursework

Luca Gough

November 2024

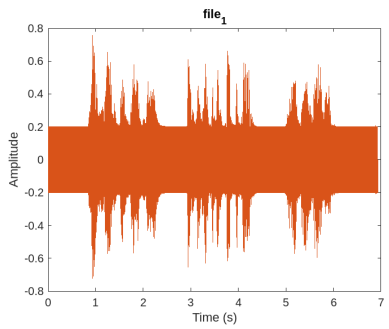
Contents

1 Audio Signal Processing

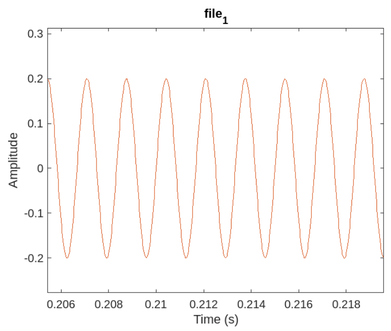
1.1 Noise Frequency Identification

The methodology used to identify the frequency of the noise component of each signal first involves plotting the amplitude over time for each signal. Zooming in on this plot in the region before the speaker starts talking, we can visually observe a sinusoid with a constant frequency. We can measure this frequency by taking the Fourier transform of the signal, which yields its frequency domain. This function describes the magnitude of the various frequency components which comprise the audio signal. A natural signal (the speaker talking) fluctuates in frequency greatly over time [1], this can be seen in figures 1c, 2b and 3c as such each signal contains many low magnitude frequencies between 100Hz and 700Hz. Since the noise remains at a constant frequency throughout the signal, its frequency will have a large magnitude in the frequency domain. This is visible in the frequency domain as a large peak. The frequency of this peak is identified by locating peaks with a magnitude over some threshold. I used a single script to de-noise the three signals, which has the advantage of being generalised to denoising any voice audio signal with a constant frequency noise component. However, using multiple scripts may have been more effective at denoising the individual signals, as it could be specialised.

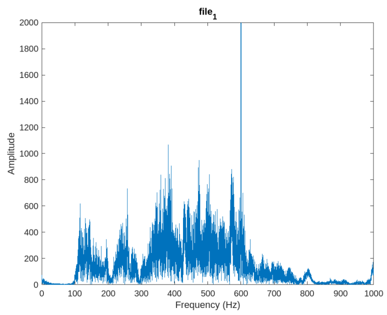
1.1.1 File 1



(a) Amplitude over time



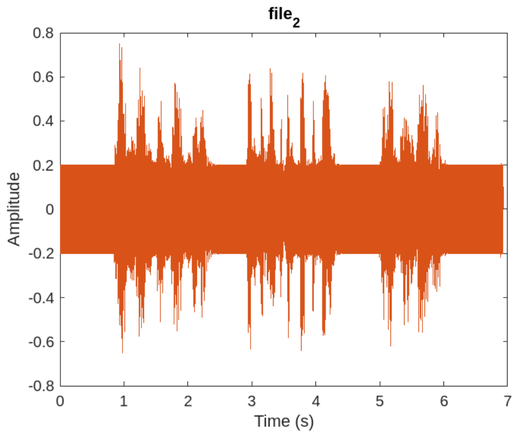
(b) 1a zoomed on the first 1s.



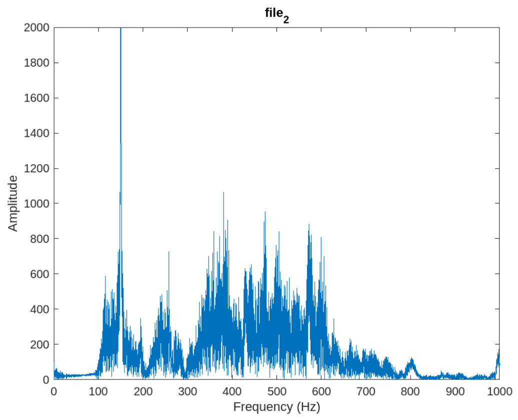
(c) Frequency Domain

The peak in the Fourier domain occurs at exactly 600.1Hz, indicating that this is the frequency of the background noise in the signal.

1.1.2 File 2



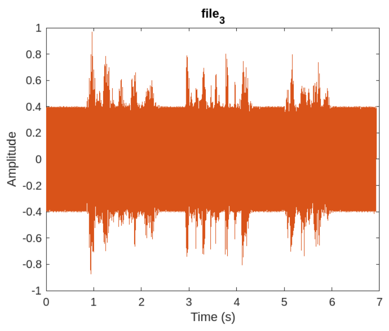
(a) Amplitude over time



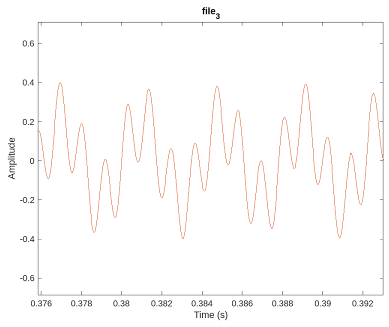
(b) Frequency Domain

The peak in the Fourier domain occurs at 150.2Hz, this is the frequency of the background noise.

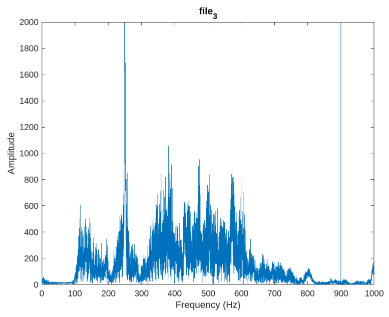
1.1.3 File 3



(a) Amplitude over time



(b) 3a zoomed on the first 1s.



(c) Frequency Domain

File 3 contains background noise with a two frequency components, we can observe these frequencies in the time domain 3b and the frequency domain 3c. They occur at 250.1Hz and 900.1Hz

1.2 Noise Removal

In order to remove the noise from each signal, we can apply some form of band-stop filter to the Fourier domain, which will remove a specific range of frequencies. In these cases the noise frequencies have very thin bandwidths and therefore a notch filter is suitable [2, 3]. Typically, they are used to remove the 50Hz power line 'hum' from various signals, which manifests in a very short bandwidth. Notch filters, as demonstrated in figure 4 converge their point of maximum attenuation on an infinitesimally small frequency value. The surrounding frequencies are attenuated inversely proportional to their distance from the target frequency. This should preserve more of the original signal than a standard bandstop filter, which may nullify frequencies surrounding the noise depending on the bandwidth parameter. This filter is parameterised by it Q, which controls bandwidth, and its 'order which determines the shape of the roll-off slope.

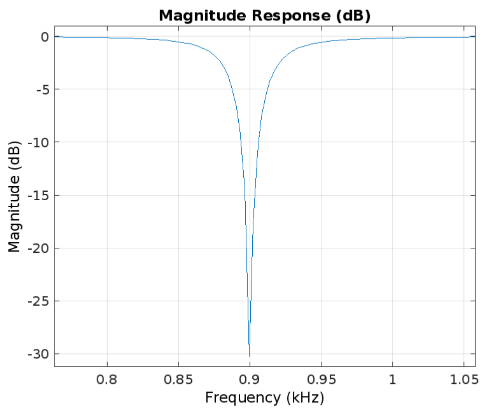
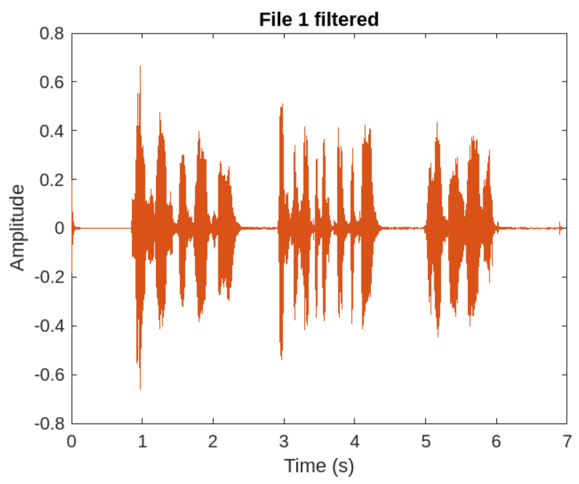
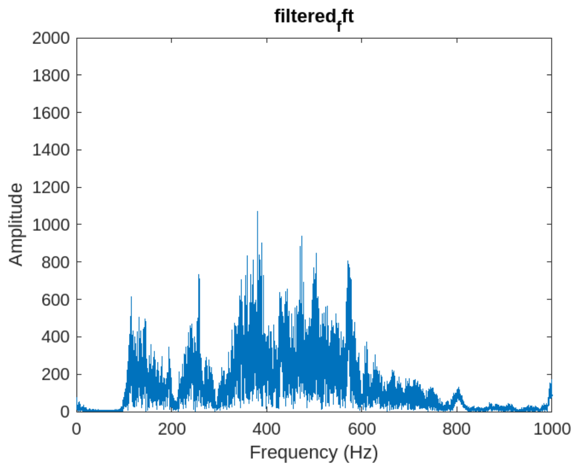


Figure 4: **Notch Filter Magnitude Response**

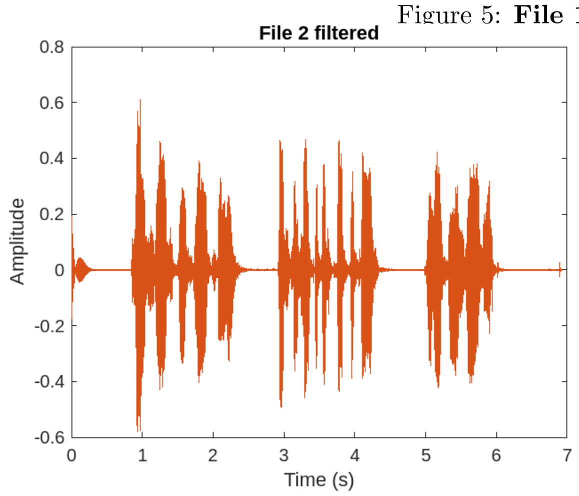


(a) Amplitude over time

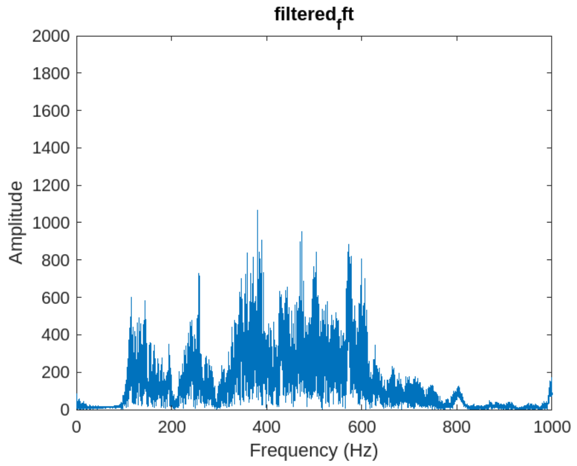


(b) Frequency Domain

Figure 5: **File 1 after denoising**

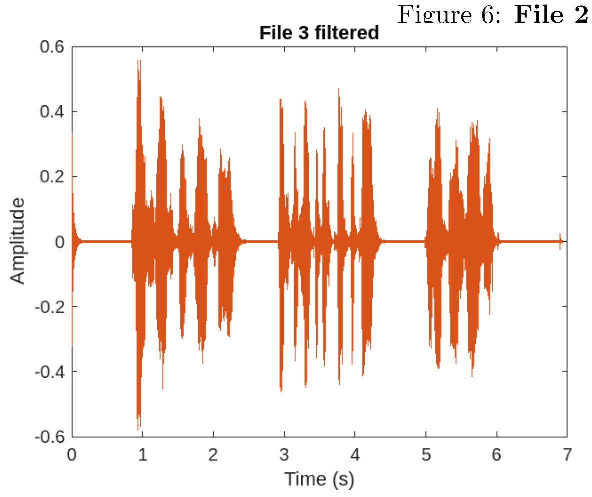


(a) Amplitude over time

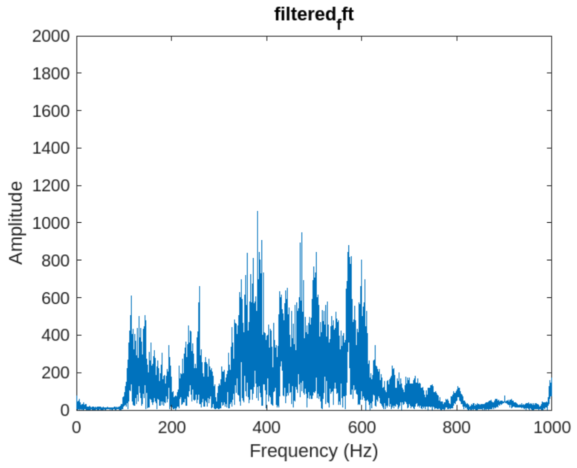


(b) Frequency Domain

Figure 6: **File 2 after denoising**



(a) Amplitude over time



(b) Frequency Domain

Figure 7: **File 3 after denoising**

Figures 5, 6 and 7 show the amplitude over time (left hand-side) and the frequency domain (right hand-side) for each signal, post-filtering. It's clear from the time-domain graphs, the constant amplitude background noise is almost entirely eliminated, indicating that peaks in the Fourier domain were in-fact the noise components. The frequency domain graphs show the regions where the notch-filter has attenuated its band of frequencies, this will inevitably remove some of the original voice signal. In terms of audible differences, file 1's high pitch sounds such as 's"s might be slightly muffled as we removed one of the higher pitch components of the signal. For signal 2 a lower frequency was removed and therefore some of the lower frequency sounds may sound more muffled. Personally, I cannot notice the difference between the two signals. There is an audible chirp at the start of each signal, this is due to the edge effect.

2 Image Analysis

2.1 Pipeline

Following the workflow outlined by [4]. The first step in designing my classifier algorithm was to examine the sample images and determine generalised the algorithm needed to be. Crucially, all the images are taken from the same angle on a grey background with good lighting. The background has some speckled noise, indicating that some level of filtering might be required. The very small dataset size meant that a conventional image analysis approach would be more suitable than a machine learning or deep learning model. One of the simplest methods for classification is segmentation, which aims to partition an image into disjoint regions of pixels. The properties of these regions can be used to classify which class a region belongs to. The algorithm uses the watershed algorithm for segmentation [5]. Zhang et. al [6] used the watershed algorithm for successfully segmenting x-ray images, extracting connected features from the background. Wen et. al [7] provides a literature review of segmentation methods from classical methods such as thresholding and the watershed algorithm to deep CNNs. This review indicates that the marker-controlled watershed algorithm can provide accurate results and actively aims to prevent over-segmentation.

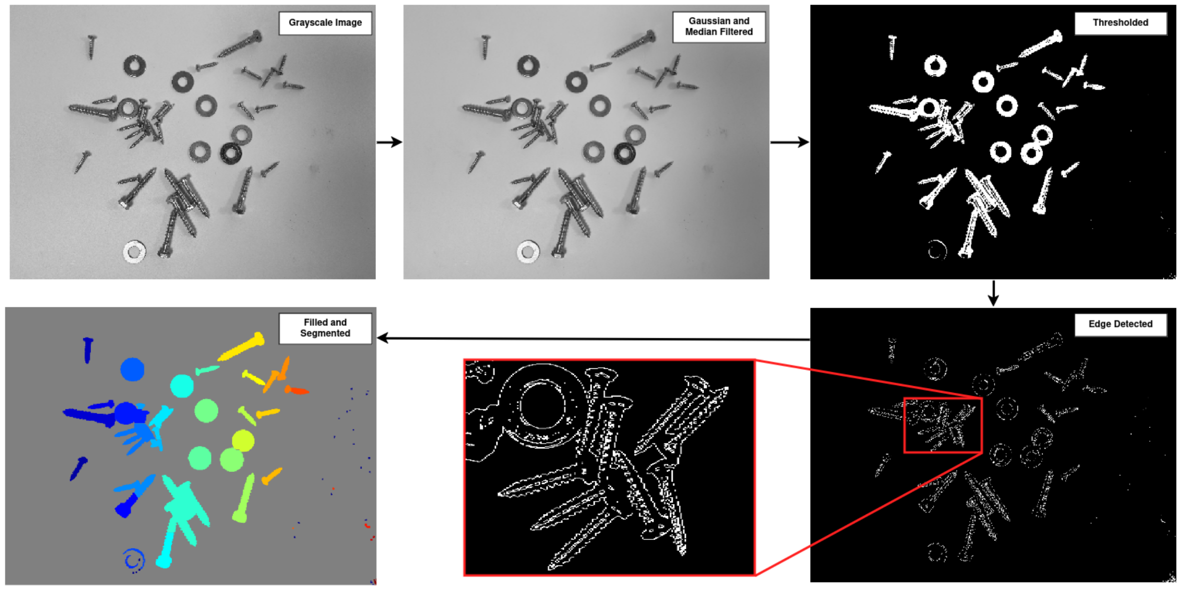


Figure 8: **Segmentation Pre-processing Pipeline**

In order to perform segmentation, we must pre-process the images to remove noise and amplify the edge contrast. In my model, we first apply a median filter, which can remove salt and pepper noise by assigning each pixel to the median value of its neighbourhood. Additionally, a Gaussian filter smooths out high frequency noise whilst preserving the contrast of the edges [8]. The next step is to threshold the image, this helps to initially segment the dark objects from the bright background. Any pixels greater than a specific threshold are set to black, and all other pixels are set to white. The watershed algorithm requires well-defined edges for accurate segmentation, so we perform an edge detection step using a "Sobel" convolution. The edge map is then morphologically dilated, which expands the border of each object, connecting any disconnected edges. This is required for the next stage, which is hole-filling. The watershed algorithm requires the foreground objects to be solid, contiguous regions separated by well-defined watersheds. For this reason, we fill in any internal gaps with a flood-fill algorithm.

The first step in the marker-controlled watershed algorithm [9, 6] is to compute the Euclidean distance transform of the image. This process replaces each pixel in the image with its distance from the nearest background pixel. Next we apply a Gaussian filter to the distance transform, this helps to smooth out any high frequency noise in the distance transform. Next, we compute the extended-maxima transform of the distance transform, this returns the points within each region which are furthest from an edge pixel or in other words the centre coordinates of the centre of each region. These markers are used as the seed points for the watershed algorithm. After segmenting the image, we can compute certain properties for each segment, such as its bounding-box, major/minor axis lengths and eccentricity. Eccentricity is one of the first metrics, in the pipeline, used to classify screws from washers. Since washers are very circular, their regions will have a much lower eccentricity than screws, which can be approximated as highly eccentric ellipses. Furthermore, long screws will have a longer semi-major axis than short screws, so we can use K-means clustering to separate the regions into the 2 categories.

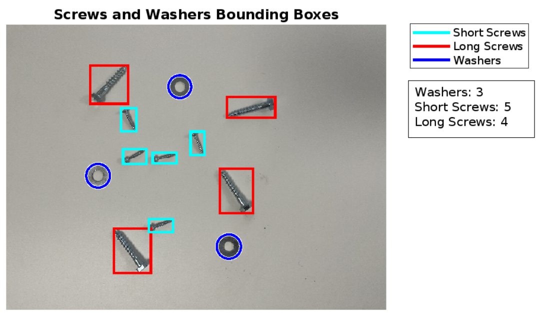
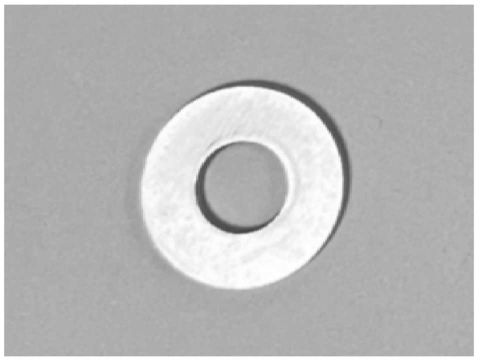
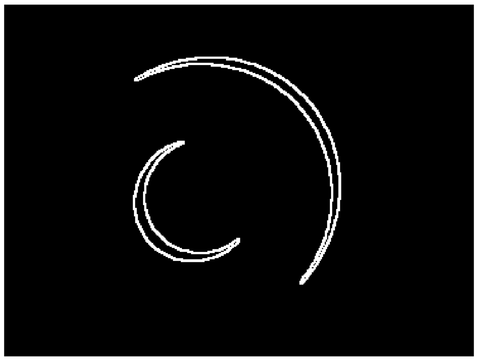


Figure 9: **Easy image, classification using segmentation only**

This method on its own works well for the easier images where the objects are not touching and have well-defined separated edges. However, it's clear from the segmentation in figure 8 that the watershed algorithm is unable to accurately segment screws which are touching along their long axis. Additionally, highly reflective washers are white enough to be thresholded as the background. This is somewhat mitigated with edge detection, as they have a dark shadow around their edge however, if we cannot detect the entire edge, this is demonstrated in figure 10. After segmentation, the two regions in figure 10b both have high eccentricities as they are not completed circles and are therefore not detected as washers.



(a) **Reflective washer, example:** High brightness foreground object.



(b) **Reflective washer after thresholding and edge detection**

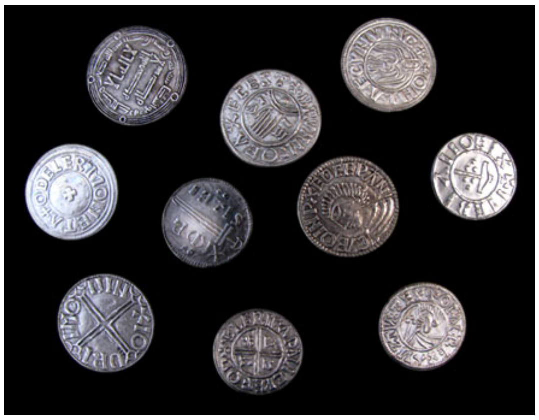
Figure 10: Reflective Washer

In order to detect these reflective washers effectively, we require a new approach, one that can extrapolate a circle from a limited set of edge points. A Hough transform seemed ideal for this scenario. As described by Richard Duda and Peter Hart in 1972 [10] the Generalised Hough Transform represents detected features as a single point in a parameter space. In the case of circles, the points in the parameter space describes a circle centre (x_0, y_0) and radius r. The general algorithm for constructing the Hough space for circles first requires computing the gradient magnitude and gradient direction $G(x, y)$ of each point in the space. This can be performed with vertical and horizontal Sobel operators. For every pixel satisfying $|G(x, y)| > T_s$ (i.e. for every edge pixel) perform the following operation:

$$\forall r \in \{r | (R_{\min} < r < R_{\max}) \wedge r \in \mathbb{N}\} \tag{1}$$

$$H(x_0, y_0, r) = H(x_0, y_0, r) + 1 \qquad \text{where} \qquad \begin{cases} x_0 = x + r\cos(\angle G(x, y)) \\ y_0 = x + r\sin(\angle G(x, y)) \end{cases} \tag{2}$$

Intuitively, this process maintains an accumulator for every possible circle with a given radius and centre location in the image, and the accumulator space for a given circle is incremented for each pixel which could exist on that circle. Peaks in the Hough space represent the circles for which the most pixels lie on.


(a) **Example image:** Circular coins on a dark background

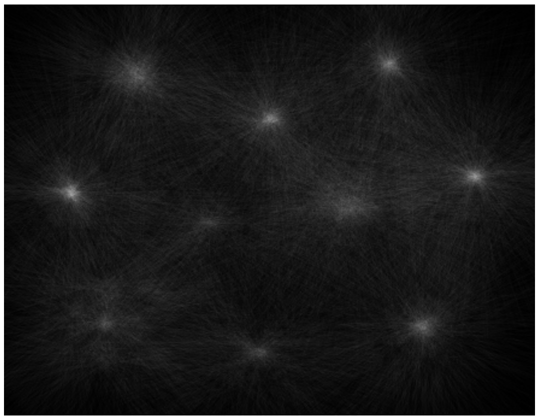
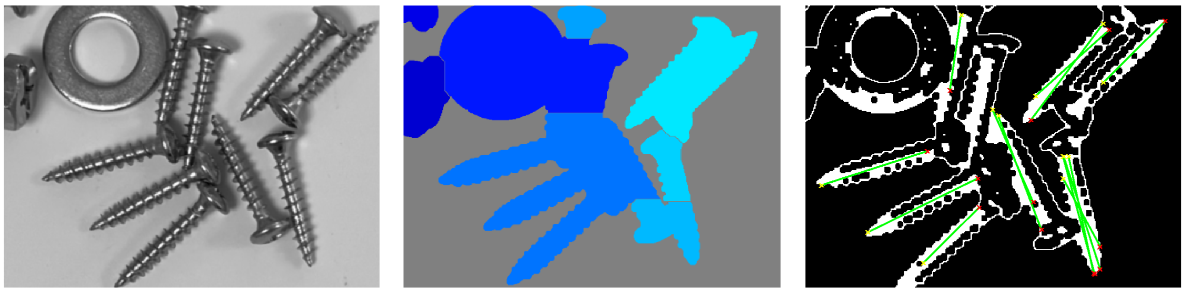

(b) **Circular hough transform:** 2D, the radius dimension is not included

Figure 11: **Circular Hough Transform:** The bright spots in the Hough Transform 11b represent the centres of detected circles in the example image 11a.

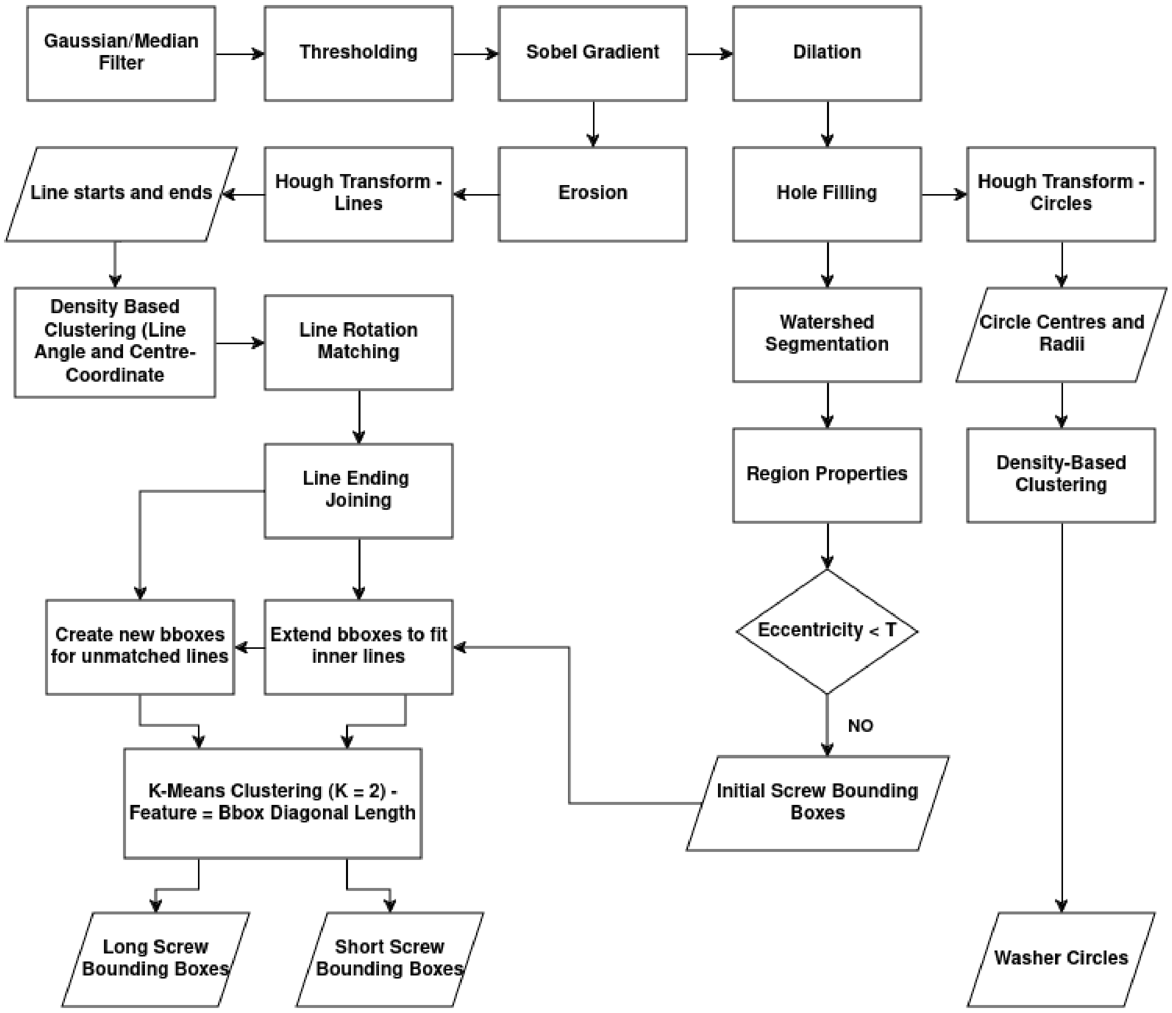
An issue generated by this approach is the Hough Transform typically detects multiple circles for each washer. To avoid overcounting, we must cluster these circles to the same point. Since we don't know how many washers are in each image, we must use a non-parameterized clustering technique, such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN). First proposed by by Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu [11]. DBSCAN assigns points to a cluster if the point is density-connected to a point within the cluster. This enables the algorithm to cluster long chains of points. However, crucially, the algorithm can produce any number of clustered defined by a maximum distance value which defines whether a point is density-connected to a cluster. DBSCAN clustering the result of the Hough Transform yields an accurate circle detector for the washers, since it can identify imperfect instances of circles.

The major issue with the watershed transform alone is if objects are significantly touching, it becomes much harder to segment them accurately. This effect is amplified by the dilation step, which expands the border of each object. In order to classify these closely touching objects/those with overlapping shadows, we augment the existing pipeline with a Hough line transform. This exploits the fact that screws are straight lines and these lines can be detected. In order to prevent extra lines being drawn between screws, we use the morphologically eroded edge map as input to the hough line transform 12c.The Hough space is parameterized by an angle and distance from the origin (the two parameters required to define any line in 2D space). The Hough Lines are also clustered using DBSCAN however they require a more complex feature set to cluster. In this case, we use the line's angle, and it's centre coordinate. After clustering, we are still left with many extra lines. These are further cleaned by joining lines which are having their ends close enough and have a similar angle. The model generates Hough lines with very wide constraints on length, so initially very short lines can be produced. After joining lines, end-to-end if their length is under some threshold they will be removed from the set.



(a) **Poor segmentation example** (b) **Under segmentation** (c) **Hough Lines**

Figure 12: **Segmentation/Hough Lines**



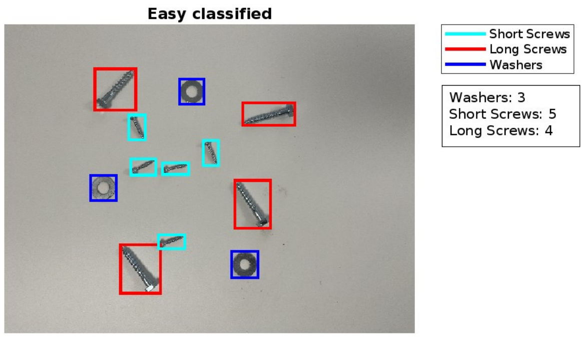
Gaussian/Median Filter
Thresholding
Sobel Gradient
Dilation
Line starts and ends
Hough Transform - Lines
Erosion
Hole Filling
Hough Transform - Circles
Density Based Clustering (Line Angle and Centre-Coordinate
Line Rotation Matching
Watershed Segmentation
Circle Centres and Radii
Line Ending Joining
Region Properties
Density-Based Clustering
Create new bboxes for unmatched lines
Extend bboxes to fit inner lines
Eccentricity < T
NO
K-Means Clustering (K = 2) - Feature = Bbox Diagonal Length
Initial Screw Bounding Boxes
Long Screw Bounding Boxes
Short Screw Bounding Boxes
Washer Circles

Figure 13: **Screw and Washer Classification Pipeline**

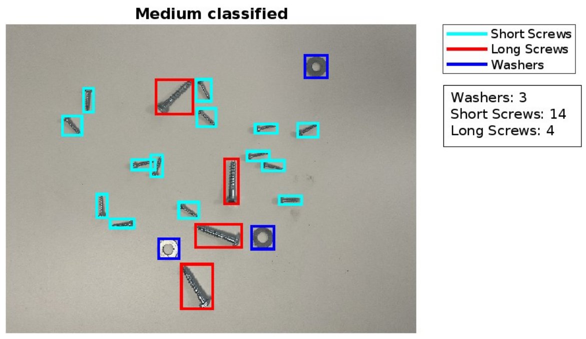
Finally, we can augment the watershed classification system to include the results of the Hough Lines. The first step is to check if any of the lines are coincident with a bounding box's diagonal detected by the watershed algorithm. In these cases, the bounding box is extended to cover the entire line. This helps to classify the cases where the watershed algorithm split a segment horizontally across a screw, which was leading to long screws being classified as two short screws. The Hough line for those long screws will extend their bounding box. For any Hough lines without a covering bounding box, we create a new bounding box. These are screws which were undetected by the watershed algorithm. Finally, the new set of bounding boxes is fed into the K-means clustering algorithm, using the diagonal length of each bounding box as the distance metric (screws lie diagonally in their bounding box). This classifies the screws into short and long screws. The full pipeline diagram can be seen in figure 13.

2.2 Results

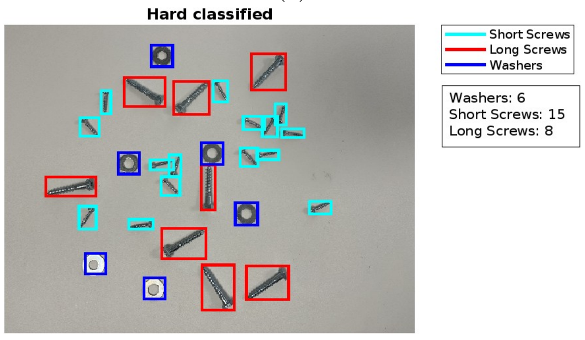
The total count of objects is simply the sum of the detections for each class: [12/12, 21/21, 29/29, 37/37, 29/40], easy-extreme. Figure 14 shows the classification results for each of the 5 test images. The model is able to achieve 100% accuracy on easy, medium and hard. On the very hard image, the model miss-classifies a single short-screw as a long screw. On the extreme image, the model's accuracy drops to 69% as it fails to identify 8 of the 21 short-screws and miss-classifies 2 as long screws. The F1-score compensates for the imbalanced class distribution, since the precision for washer identification across all images is 100%, and its recall is 96% this increases the F1-score for all images.



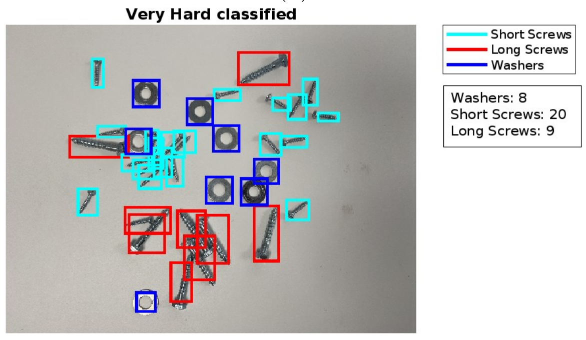
(a)



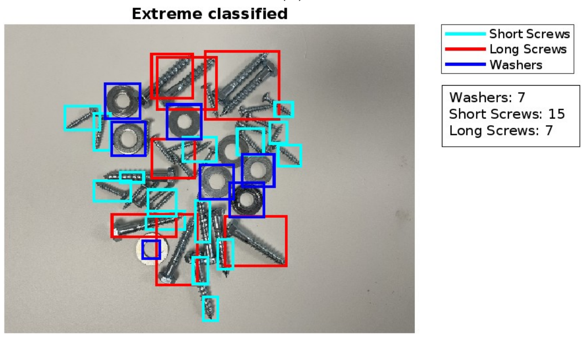
(b)



(c)



(d)



(e)

Image	Accuracy Score	F1 Score
Easy	100%	1.00
Medium	100%	1.00
Hard	100%	1.00
Very Hard	98%	0.97
Extreme	69%	0.74

(f) **Accuracy and F1-scores**

Figure 14: **Classifier results [easy-extreme]**

In order to measure the accuracy and F1 scores for the detector, I had to manually identify the ground truth bounding boxes. The program then uses the thresholded intersection over union (IOU) to match the predicted bounding boxes with the ground truth bounding boxes. These pairs can be used to compute the recall and precision of the detector. IOU measures the ratio of overlap and the total area between two bounding boxes [12]. The loss in accuracy and F1-score in the extreme image is due to the increased number of false negatives. This is the result of poor segmentation, since many of the objects are touching in several places. Additionally, the parallel screws lead to under-fitting of the Hough line clustering (the two lines identified for each screw are aggregated in a single cluster). Furthermore, short-screws which are positioned end-to-end have their hough lines joined end-to-end and are detected as a long-screw.

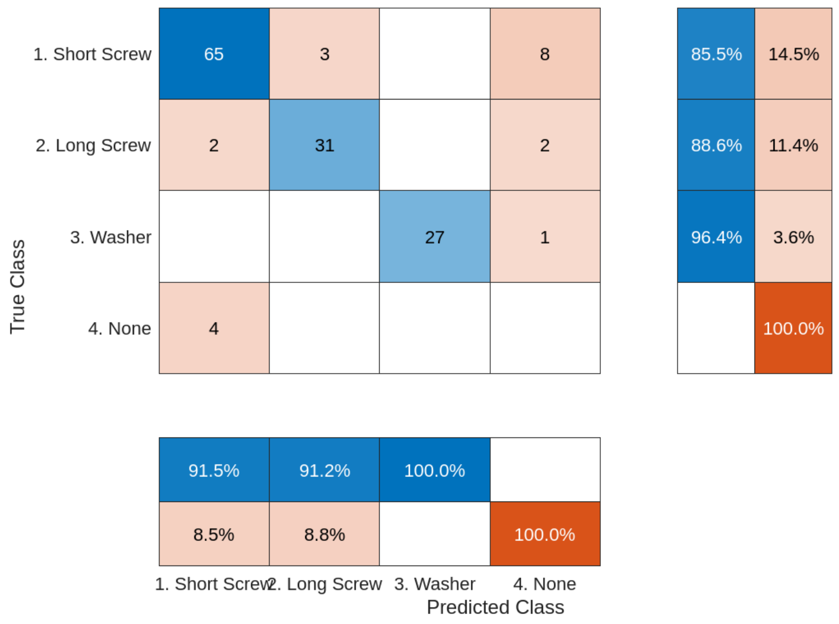


Figure 15: **Confusion Matrix (All images)**

The full confusion matrix for the algorithm applied to all images can be seen in figure 15, demonstrating that most of the loss in accuracy comes from false negatives. Additionally, the model is most accurate at detecting washers, this is due to the robustness of the Hough circle algorithm, accurately detecting partial circles. Short screws are the most miss-classified and undetected, this is mainly affected by the magnitude of the morphological erosion used at the start of the Hough lines pipeline. Under-erosion will fail to disconnect the borders of touching screws, generating erroneous Hough lines. Over-erosion disconnects edges such that lines can no longer be detected. A potential way to fix this issue would be to use the thresholded distance transform, used for the watershed algorithm, as input to the Hough lines algorithm. If tuned correctly, it could potentially highlight a line of pixels down the centre of each screw, which would be disjoint from the lines of other screws.

Another potential improvement could be to use a Viola-Jones detector [13]. This approach optimises a set of weights which comprise a weighted summation of Haar-like features. Thresholding this weighted sum can be used to classify areas within images containing specific features. Haar-like compute the difference in summation of adjacent regions within a detection window. Original proposed as a facial recognition algorithm, it is often adapted for detecting objects. This could be used to identify the bounding boxes of screws within images, and using the existing k-means classifier to differentiate between long and short screws. The original paper achieved a 71% accuracy on their facial recognition detector, however used in conjunction with the existing pipeline, it could potentially identify some of the short-screw false negatives. Further optimisations of the segmentation may also improve the F1-scores. For this project I opted to let the watershed algorithm under-segment and detect the undetected objects with other algorithms rather than over-segment and generate many false positives. The following paper [14] demonstrates the use of clustering on the segments produced by the watershed algorithm to reduce over-segmentation in medical imagery. They report a reduction of segmentation by up to 94% using clustering.

References

[1] J. Mahdi, "Frequency analyses of human voice using fast fourier transform," *Iraqi Journal of Physics (IJP)*, vol. 13, pp. 174–181, 02 2019. 1

[2] C. S. Analysis. [Online]. Available: https://resources.system-analysis.cadence.com/blog/msa2021-understanding-a-notch-filter-transfer-function 2

[3] [Online]. Available: https://www.analog.com/en/resources/glossary/notch-filter.html 2

[4] C. Byrne, "Development workflows for data scientists," O'Reilly Media. Inc., Tech. Rep., 2017. 4

[5] S. Beucher and C. Lantuéjoul, "Use of watersheds in contour detection," *In International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation*, vol. 132, 01 1979. 4

[6] X. Zhang, F. Jia, S. Luo, G. Liu, and Q. Hu, "A marker-based watershed method for x-ray image segmentation," *Computer Methods and Programs in Biomedicine*, vol. 113, no. 3, pp. 894–903, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016926071300415X 4, 5

[7] T. Wen, B. Tong, Y. Liu, T. Pan, Y. Du, Y. Chen, and S. Zhang, "Review of research on the instance segmentation of cell images," *Computer Methods and Programs in Biomedicine*, vol. 227, p. 107211, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260722005922 4

[8] R. Gonzalez and R. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, 1992. 4

[9] R. Gaetano, G. Masi, G. Scarpa, and G. Poggi, "A marker-controlled watershed segmentation: Edge, mark and fill," 07 2012, pp. 4315–4318. 5

[10] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, p. 11–15, Jan. 1972. [Online]. Available: https://doi.org/10.1145/361237.361242 6

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, E. Simoudis, J. Han, and U. M. Fayyad, Eds. AAAI Press, 1996, pp. 226–231. [Online]. Available: http://dblp.uni-trier.de/db/conf/kdd/kdd96.html#EsterKSX96 6

[12] S. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. D. Reid, and S. Savarese, "Generalized intersection over union: A metric and A loss for bounding box regression," *CoRR*, vol. abs/1902.09630, 2019. [Online]. Available: http://arxiv.org/abs/1902.09630 9

[13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I. 9

[14] H. P. Ng, S. H. Ong, K. W. C. Foong, P.-S. Goh, and W. L. Nowinski, "Medical image segmentation using k-means clustering and improved watershed algorithm," *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 61–65, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:17858176 9