Automatically Masking Cartridge Case based on Mask R-CNN

Ziqi Dou, M.A.Sc, M.Eng

Abstract

The report begins by providing a brief overview of the markings found on discharged cartridge cases. It then introduces the Mask RCNN algorithm and explains how it is employed to train and automatically apply masking to two datasets. The first dataset consists of 83 fired cartridge case images sourced from Google Images with random calibres, while the second dataset comprises 227 images from the NBTRD (NIST Ballistics Toolmark Research Database) [4], where these images closely resemble the 3D microscope images mentioned in the assignment requirements and offer a more realistic representation of forensic science scenarios. The dataset covers 9mm, 38/357, and 40/10 mm calibers. The report also covers the preprocessing, training, and the discussion of the automatic masking results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As shown in Figure 1.1. (a) When the firing pin hits the primer, the powder is ignited and propels the bullet out of the barrel. At the same time, the explosive force pushes the cartridge case against the breech face of the firearm. (b) The lands and grooves fabricated in the barrel leave striated toolmarks on the bullet when it passes through the barrel. (c) The Firing pin, Breech face and Ejector of a firearm. (d) Three corresponding impressed toolmarks on the cartridge case. (e) Striated toolmarks on the bullet [6].

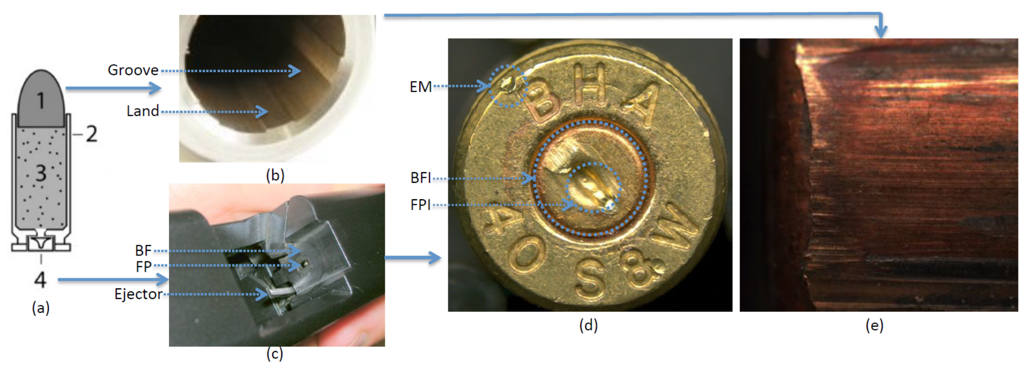


Figure 1.1: The formation and taxonomy of ballistic toolmarks [6]

Variability in breech-face and firing-pin impressions across different firearms introduces distinct patterns, making it possible to infer which firearm fired a particular cartridge case based on the observed irregularities [1]. This information may assist forensic practitioners in their work.

1.2 Objective

The forensic laboratory utilizes a 3D microscope to capture images of fired ammunition components, particularly cartridge cases, for investigative purposes. These images are processed by software to determine whether the cartridges originated from the same or different firearms, providing crucial information for police investigations.

Currently, the preparatory function of masking, aimed at aiding the software in identifying specific features on the cartridge case, is carried out manually. This manual process is laborious and time-consuming, necessitating an automated solution for efficiency. The requirement is to develop an algorithm capable of automatically masking cartridge case images, streamlining and enhancing the overall forensic analysis workflow.

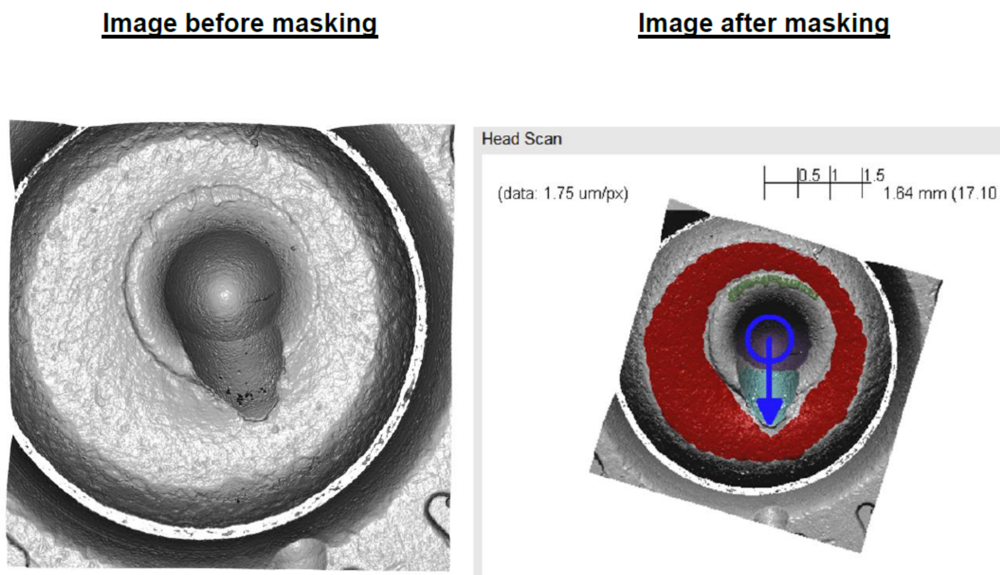


Figure 1.2: Example of masking

Chapter 2

Methodology

2.1 Data Collection

Two datasets were used in the assignment. The first one was from Google Images with a random selection regardless of specific calibres, and 84 fired cartridge case images in total. Each image may contain more than one cartridge case, as shown in Figure 2.1. Most of the cartridges in this dataset did not have clear firing pin drag or flowback (The explosion can push outward the area around the firing-pin impression leads to convex deformation is known as flowback [1].).

Figure 2.1: .45 calibre cartridges

Calibres	Number of Images
Random	83
Total	83

Table 2.1: Number of Internet Images

The second dataset was downloaded from the NBTRD Ballistics Toolmark Database [4], where these images closely resemble the 3D microscope images mentioned in the assignment requirements and offer a more realistic representation of forensic science scenarios. The dataset covers 9 mm, 38/357, and 40/10 mm calibres as shown in Table 2.2. There are 227 images in total and each image only contains one cartridge case with a clear firing pin impression, firing pin drag, flowback and breech face impression, as shown in Figure 2.2.

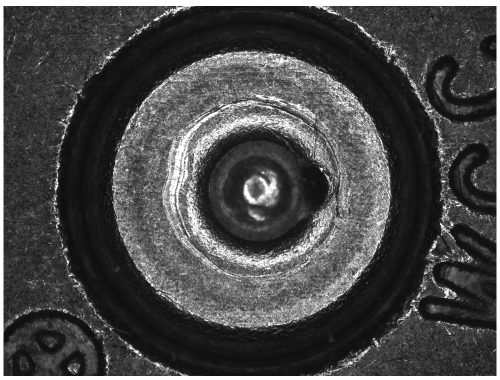


Figure 2.2: 9 mm calibre cartridge

Calibres	**Number of Images**
9 mm	208
40/10 mm	7
38/357	12
Total	227

Table 2.2: Distribution of calibres

2.2 Mask R-CNN model

In recent years, the Mask R-CNN (Region-based Convolutional Neural Network) algorithm has emerged as a state-of-the-art technique for object instance segmentation. Figure 2.3 is an implementation of Mask R-CNN. The model generates bounding boxes and segmentation masks for each instance of an object in the image.

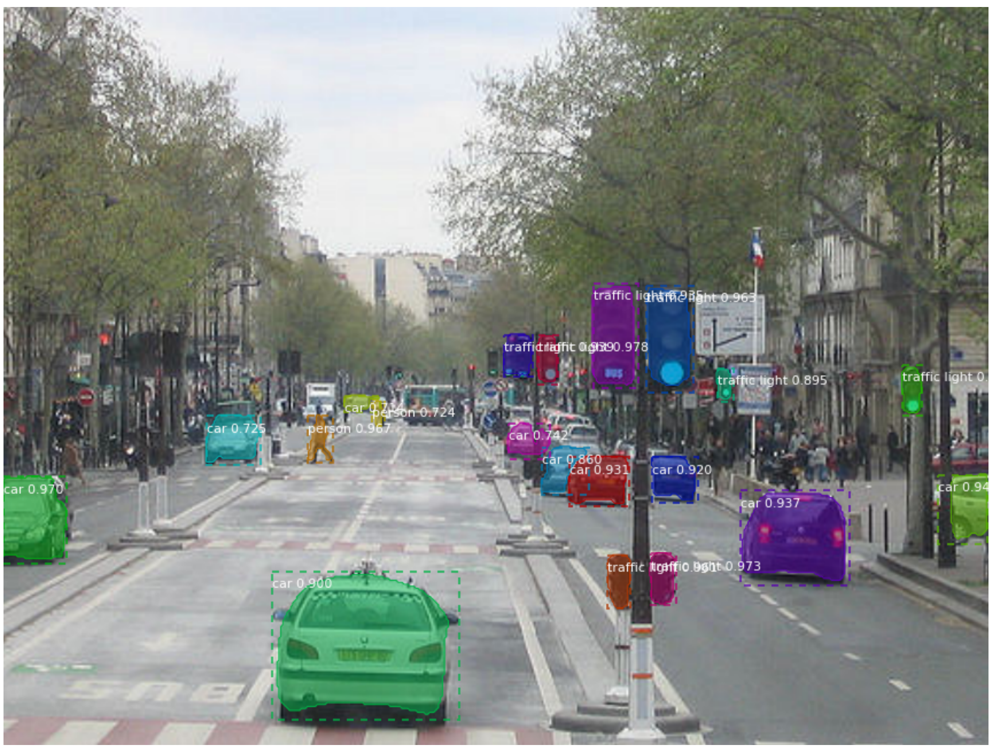


Figure 2.3: Object Detection and Segmentation [3]

More details about Feature Pyramid Network (FPN) and ResNet101 backbone used in Mask R-CNN were introduced in [2]. This assignment used the algorithm as a base skeleton to detect and segment a cartridge's firing pin impression, drag, and breech face. The used packages and the created scripts are listed in the Appendix. Version control information is listed in the 'requirement.txt' file.

Chapter 3

Implementation

3.1 Preprocessing

The preprocessing of the image datasets is the most time-consuming process. The following steps were applied:

a. Images with resolutions and dimensions that are too small or too large were excluded.

b. Changed image names, ensuring they can be imported and read by the program, and standardized the file extensions of the images to either '.jpg' or '.png'.

c. Used 'labelme' to manually segment the regions of interest. (Note: 'labelme' may not be compatible with Python 3.7. Python 3.9 was utilized for executing 'labelme' in the assignment.)

And then, the following regions were segmented by using 'labelme':

1. The firing pin impression

2. The firing pin drag if present

3. The breech face impression excluding flowback

4. The direction of firing pin drag cannot be segmented, but it can be calculated from the bounding box information.

5. Due to my limited knowledge and experience, the aperture shear has not been marked.

Two labelled image examples are shown in Figure 3.1 and Figure 3.2 respectively. The firing pin impression (green), firing pin drag (brown) and breech face impression (red) were labelled in both images. According to Ott *et al.*[5] and Song *et al.*[7], flowback has usually been excluded from analysis. Therefore, the observable flowback is also excluded in the second image. Their labelled information is stored as '.json' files in the folder 'imageNew' and folder 'image' respectively.

Figure 3.1: Internet image

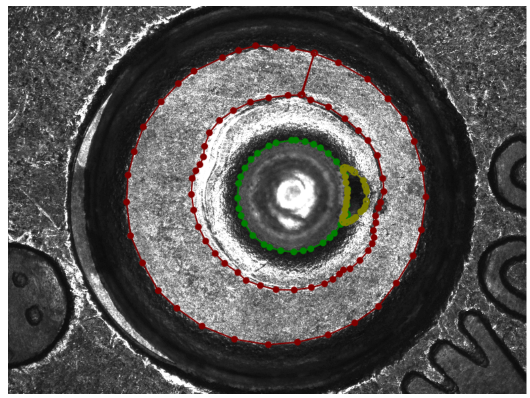


Figure 3.2: Image from NBTRD

3.2 Training

A typical machine-learning methodology was followed. Each dataset was divided into three parts: Train set, Validation set and Test set. Their distributions are listed in Table 3.1 and Table 3.2.

Calibres	**Train**	**Validation**	**Test**	**Total**
9 mm	135	34	39	208
40/10 mm	4	2	1	7
38/357	4	3	5	12
NBTRD Images	**143 (63%)**	**39 (17%)**	**45 (20%)**	**227**

Table 3.1: Distribution of NBTRD Images

	Train	**Validation**	**Test**	**Total**
Internet Images	**62 (75%)**	**10 (12%)**	**11 (13%)**	**83**

Table 3.2: Distribution of Internet Images

The iteration process involves training the model through multiple iterations (epochs) using the training datasets. In each iteration, the model processes batches of training samples, computes the loss and performs back-propagation to update the model parameters. The training process aims to minimize the overall loss, enabling the model to accurately detect objects, classify them, and segment instance masks.

This iterative training continues until the model reaches a predefined number of epochs. The 'Steps_Per_Epoch' was set to 50, 'Epoch' was set to 100 and 'batch size' was set to 64 initially. The updated parameters are stored in the 'logs' folder.

Chapter 4

Results

This section showcases the automated masking results and illustrates how the loss function converges with the progression of epochs. The full results are stored in the folder 'imagesNew/ testresult' and the folder 'images/testresult' respectively. Configuration information is listed in Table 4.1.

Table 4.1: Hardware and Software Configuration

Hardware	
Processor	Intel Core i7-9750H @ 2.60GHz
Memory	32 GB DDR4 @ 2667 MHz
Graphics	NVIDIA RTX2080 Max-Q 8GB (Compute Capability: 7.5)
Software	
Python version	3.7
TensorFlow-GPU Version	1.14
CUDA Version	10.0
cuDNN Version	7.4

After the training process, the updated parameters are stored in the folder 'logs' with extension '.h5'. The loss function curves can be monitored and checked in Tensorboard (http://127.0.0.1:9090 in local browser) after terminal cmd 'tensorboard –host 0.0.0.0 –logdir ./logs/ –port 9090'. The loss function curves are stored as CSV data and were processed in MATLAB, as shown in Figure 4.1 and Figure 4.3.

Internet Image Results

Due to the diversity and complexity of categories in Internet Images, the training process required a more extended duration. An interim evaluation was conducted every 100 epochs to inspect the results of automatic masking. The entire training regimen spanned 300 epochs, with a total duration of approximately 23 hours.

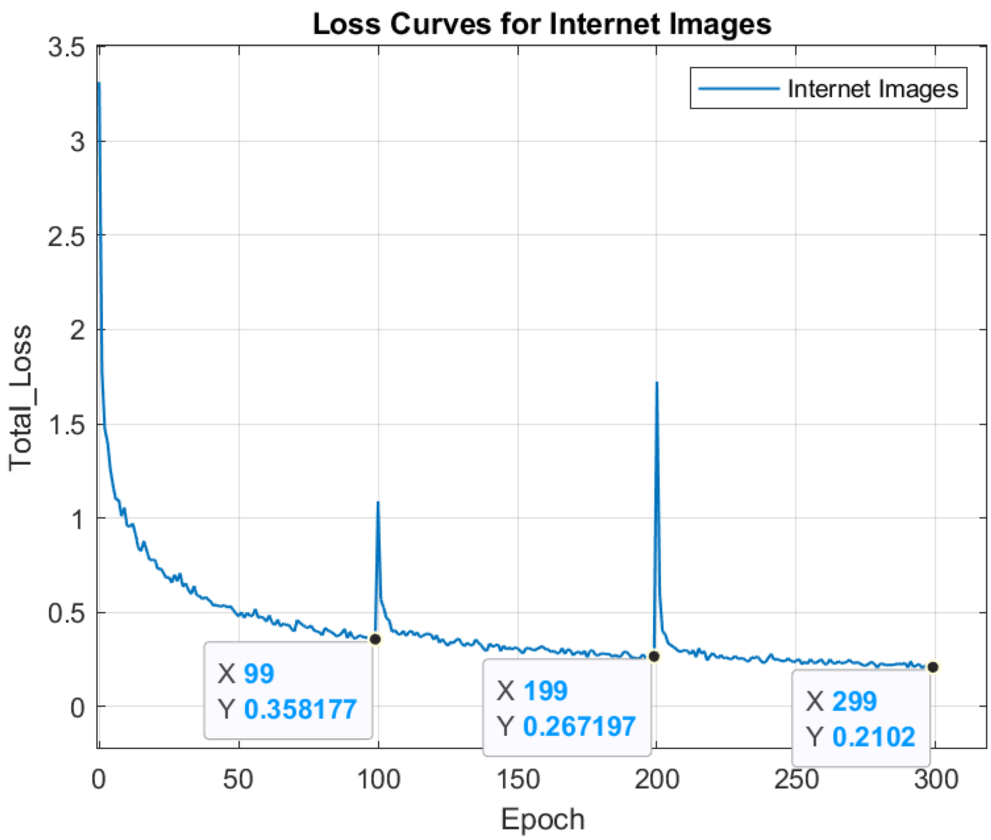


Figure 4.1: Internet images total loss curve

Despite the loss function not reaching complete convergence in the final epochs (Figure 4.1), its progression became gradual.

The automatic masking results are shown in Figure 4.2. Before masking, there are firing pin impressions, firing pin drags and breech faces in the picture. And there is no observable flowback. However, only firing pin impressions and breech faces are marked after auto masking. It may be due to the insufficient number of images for training.

(a) before masking (b) after masking

Figure 4.2: automatic masking results for Internet Image (.45)

NBTRD Image Results

Compared to Internet Images, the image dataset of NBTRD exhibited a better-converged curve after 100 epochs, with a total duration of approximately 7 hours, as shown in Figure 4.3.

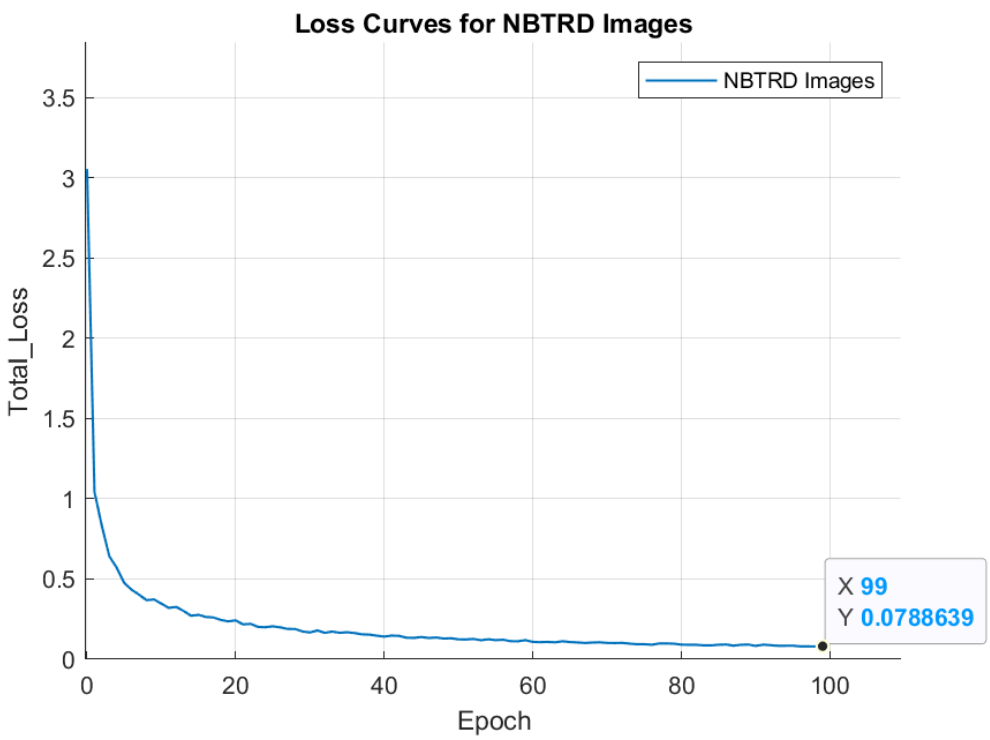
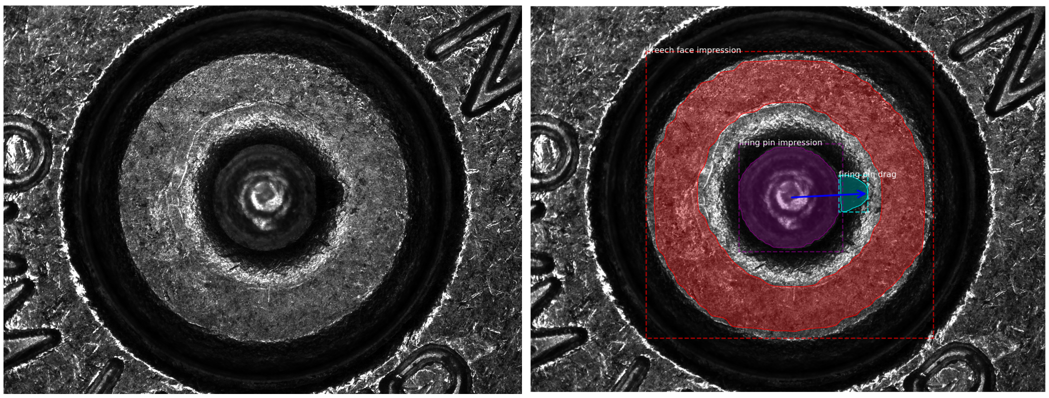


Figure 4.3: NBTRD images total loss curve

A larger number of images in the training set ensured improved automatic masking results.

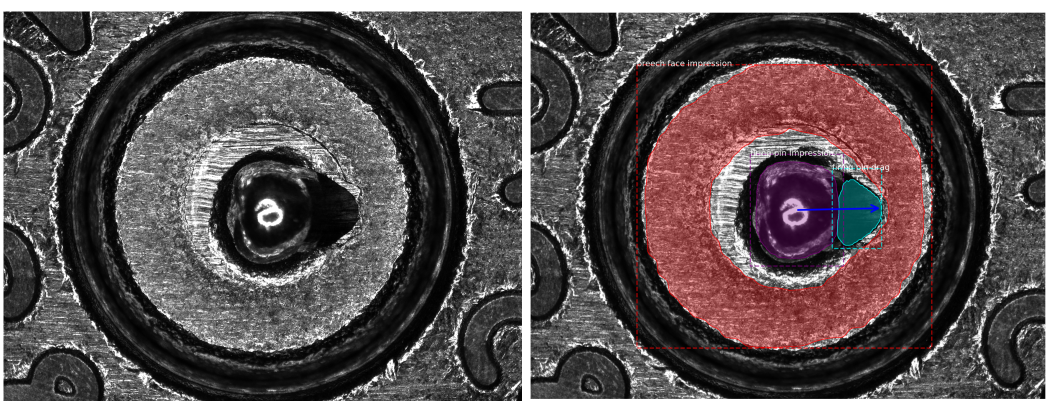
As demonstrated in Figure 4.4, the first image shows ROIs (region of interest), including firing pin impression, breech face impression, and firing pin drag. After automatic masking, the corresponding areas, including the direction of the firing pin drag, are successfully labelled. Moreover, the flowback area as a non-ROI, is appropriately excluded.



(a) before masking (b) after masking

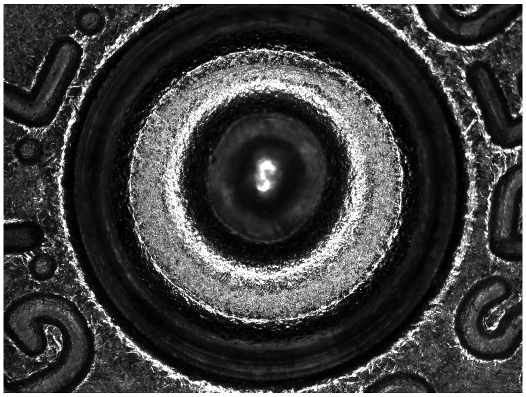
Figure 4.4: automatic masking results for NBTRD Image (9 mm).

Despite the very small size of the firing pin drag in the above images, it has been successfully masked.

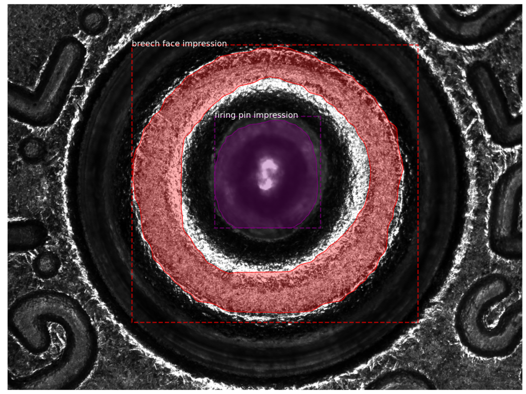


(a) before masking (b) after masking

Figure 4.5: automatic masking results for NBTRD Image (40/10 mm)



(a) before masking



(b) after masking

Figure 4.6: automatic masking results for NBTRD Image (38/357)

Likewise, aside from the 9mm calibre, the 40/10 mm and 38/357 calibre cartridge cases also demonstrate good auto masking results in Figure 4.5 and 4.6. And there is no observable firing pin drag in the later ones. Full test results are stored in the folder 'imagesNew/testresult' and the folder 'images/testresult' respectively.

Chapter 5

Discussion

In conclusion, the modified Mask R-CNN algorithm demonstrates commendable automatic masking performance on the NBTRD image dataset (9mm, 40/10mm, and 38/357). Besides automatically masking firing p in impressions, firing pin drags, and breech face impressions (excluding flowback), the enhanced algorithm efficiently computes the direction of firing pin drag from bounding box information and annotates it on the images. It is reasonable to anticipate similarly favourable outcomes when training on a dataset consisting of 3D microscope images.

Appendix A

Appendix

- `gpu_info.py`
  Retrieves all available GPU information.

- `getdata.py`
  Converts annotation data from a specific format (.json) to VIA (VGG Image Annotator) format.

- `automasking.py`
  Operates the auto-masking process with updated parameters.

- `samples/CartridgeCase/cartridgecase.py`
  The main entry of the modified algorithm.

- `mrcnn/visualize.py`
  Overrides 'display_instances' function, enabling calculation of firing pin direction and fixed mask colours.

- `logs`
  Stores the updated parameters (only the latest parameters are kept due to large disk usage).

- `image` and `imageNew`
  Consists of a train JSON set and a val JSON set.

- `images` and `imagesNew`
  Consists of a train set, a val set, a test set, and a test result set.

- `requirement.txt`
  List of version control information.

Bibliography

[1] Nabanita Basu, Rachel S Bolton-King, and Geoffrey Stewart Morrison. Forensic comparison of fired cartridge cases: Feature-extraction methods for feature-based calculation of likelihood ratios. *Forensic science international: Synergy*, 5:100272, 2022.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[3] Matterport. Mask_RCNN. `https://github.com/matterport/Mask_RCNN`. Accessed: Jan 10th, 2024.

[4] National Institute of Standards and Technology. Ballistics toolmark database, 2024.

[5] Daniel Ott, Robert Thompson, and Junfeng Song. Applying 3d measurements and computer matching algorithms to two firearm examination proficiency tests. *Forensic science international*, 271:98–106, 2017.

[6] Joseph Roth, Andrew Carriveau, Xiaoming Liu, and Anil K Jain. Learning-based ballistic breech face impression image matching. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8. IEEE, 2015.

[7] John Song, Theodore V Vorburger, Wei Chu, James Yen, Johannes A Soons, Daniel B Ott, and Nien Fan Zhang. Estimating error rates for firearm evidence identifications in forensic science. *Forensic science international*, 284:15–32, 2018.