Automatic Bloodstain Pattern Analysis
SENG402 Software Engineering Research Project

Claire Barnaby
52159849
cba62@uclive.ac.nz

27 September 2018

Abstract

Forensic science can become more objective through automatic analysis of bloodstain spatter patterns. This includes the automatic segmentation of bloodstains, as well as the classification of the bloodstain spatter pattern type. It allows analysts to quickly look at many metrics and compare their classification to the classification of the computer program. Adaptive thresholding followed by a dilation morphological operation was used to segment the bloodstains and resulted in all bloodstains being found. However, the segmented bloodstains included false positives of any markers that are not circular and rulers in the image. Six bloodstain metrics, gamma, direction, solidity, circularity, intensity, and area, were computed for each segmented bloodstain. Then three pattern metrics, convergence, linearity, and distribution, were computed. To classify the bloodstain spatter pattern, pointnet++ was trained on the points from the boundaries of the segmented bloodstains. This resulted in an accuracy of 87%, but it failed to learn the difference between impact and expirated patterns. Transfer learning with Resnet trained on images cropped around the highest density region was more successful with an accuracy of 92%. It is shown that data mining from images of blood spatter patterns is successful and that there is promise in using deep learning techniques to classify the blood spatter pattern images. The bloodstain segmentation application is ready to be used in the research of bloodstain spatter pattern analysis.

Contents

1 Introduction

In forensic science, blood spatter patterns are analyzed to determine the cause of blood in a crime scene. The analysis of these bloodstain spatter patterns is an ongoing area of research, with researchers trying to find key characteristics distinguishing the types of patterns. There are six main types of spatter bloodstain patterns: gunshot, cast off, arterial spray, expirated, impact, and drip. There are also non-spatter bloodstain patterns, such as the transfer pattern, which will not be covered in this project [8].

This project is in conjunction with the Institute of Environmental Science and Research (ESR) bloodstain pattern analysis group. They are interested in whether a computer-based system could make classifying different bloodstain spatter patterns more objective. Currently, the process of determining the bloodstain pattern type is subjective as bloodstain pattern analysts do not know which measurable characteristics define each pattern type. An automated system will enable researchers to look at more characteristics of a pattern. This is because they will be able to measure all the individual stains. This will produce more data that they can use for analyzing the patterns so they can determine which characteristics are important for pattern classification. A computer program could also tell the analyst which pattern type the bloodstain spatter pattern is. This will remove the bias of the analyst and give consistent results. The analyst will then be able to compare their result with the classifier computer program.

This will be achieved through the use of computer vision techniques such as thresholding and neural networks. Firstly, the bloodstains will be segmented from the image using thresholding techniques. Then the metrics, as decided by blood spatter pattern analysts, will be calculated from the segmented bloodstains. Finally, using the segmentation results and the images, deep neural networks will classify which type of bloodstain spatter pattern the pattern is.

2 Background

2.1 Subjectivity in Bloodstain Spatter Pattern Analysis

Bloodstain pattern analysis is part of forensic science. It refers to the interpretation of the shape and distribution of bloodstains at a crime scene. These stains are analyzed to distinguish the type of the pattern the bloodstains form [20, 8]. The patterns are difficult to distinguish between and there is an active area of research on which characteristics to measure to classify between similar patterns. Using current methods bloodstain pattern analysts get the pattern classification wrong 13.1% of the time on hard surfaces and 23.4% of the time on fabric surfaces [25]. This is partly due to the subjectivity of bloodstain pattern analysis stemming from with uncertainty to which characteristics are important [2].

Recently the field has been criticized for not having enough science in it. This criticism is important as bloodstain pattern analysis is used as evidence to convict felons. In a report about forensic sciences and their subjectivity it was stated that there are enormous uncertainties associated with bloodstain pattern analysis, due to the emphasis on experience over scientific foundations, however, some scientific foundations do exist [10]. This has a consequence of the wrong persons being convicted such as the case of Joe Bryan. Who was convicted based on evidence given by a bloodstain pattern analyst who was not sufficiently trained [9]. Which highlights the importance of science-based bloodstain pattern analysis in criminal investigations.

Automatic bloodstain pattern analysis and classification will make pattern classification more objective. In research, the analyst will be able to get data from every bloodstain and compute difficult pattern wide metrics such as the distribution easily. Such data is difficult to collect manually because

the researcher will have to measure every bloodstain manually. This is fine for patterns with a small number of bloodstains but some patterns have thousands of stains. An analyst can not accurately count that many stains let alone afford the time to measure each one. Furthermore, some bloodstains are only a couple of millimeters long so it is difficult to measure them without a large error. These measurements can be done with a computer assuming the bloodstain segmentation is accurate. It will be able to measure all the bloodstains and pattern metrics quickly which results in better data for an analyst to use. They will then be able to make a more informed decision. Automatic classification acts as the role of the analyst so it removes the subjectivity of the analyst, and only uses the data given making it more objective.

2.2 Previous Work

Currently, researchers have tried some applications such as ImageJ to get metrics on stains. This is a generalized tool used for segmenting medical images. It is used for simple image analysis, for example, to count the number of bloodstains or area analysis [38]. While there are plugins available for image J there are none that have all the analysis tools needed for bloodstain spatter analysis [34]. Subsequently, there is not a complete automated bloodstain pattern analysis tool available which means specialized characteristics cannot be analyzed.

Development of such a tool would involve continuing the work made by Ravishka M. Arthur [3]. Who developed an application that analyzed stains from cast off and impact patterns. This method used triangle thresholding to segment bloodstains on a white background and produce a binary image. The image was then analyzed with 12 features [3]. This work was done in conjunction with ESR. However, ESR cannot use this program because they can not get it to run and it is heavily tailored to the images produced for her research. ESR would like to extend this application and make it user-friendly so it could be used generally in blood spatter pattern research.

One limitation of Ravishka's method is the thresholding of the images. This is because the threshold value was only calculated once and then the value was fixed for all images. Which means that is doesn't work well for images with varied lighting or shadows and, images which do not have a skewed intensity histogram. Once the image was in a binary format the tail of the bloodstain was removed through an erosion operation and then a dilation operation. This relies on the tails to be thin enough that they are eroded away. This is a problem as small stains are eroded away so stains are lost and larger stains with thick tails do not have their tails completely removed. This is seen in her segmentation of the impact patterns as many stains are missed which may also be due to poor thresholding.

Another limitation was on how it removed the stickers used to stitch images together. When forensic analysts generate the blood spatter images they take multiple images and stitch them together to get enough detail in the image resulting in the large image. The images are stitched together using pins or markers stuck on the image which need to be removed in segmentation. In this solution, the markers are removed in the thresholding of the image. Which means that it is not robust against different marker types in other images, as it is reliant on that the markers are not within the same HSV range that they are thresholding to.

In bloodstain pattern analysis ellipses are fitted to the bloodstains in order to determine the width to length ratio which infers the impact angle of the droplet. Therefore, ellipses were fitted to each segmented stain. Then the Matlab function regionprops to gather the measurements of each stain and used this information to extract other features about the stain [29].

The features analyzed where tail to body ratio, width to length ratio, area of element to area of convex hull ratio, perimeter of inscribed circle to perimeter of element ratio, element area, element intensity, orientation of elements, standard deviation of orientation angle, proportion of intersecting

major axes, linearity, circularity of convex hull and element density [3]. Some of these twelve features were found to not be useful in the analysis so will not be implemented in this project. The features that were not useful are perimeter of an inscribed circle to the perimeter of an element, tail to body ratio, the proportion of intersecting major axes and circularity of convex hull. This was confirmed with Michael Taylor the domain expert in this project.

Ravishka M. Arthur improved on the implementation of K. Boonkhong et al [23]. They, isolated one blood stain by determining the colour of the stain and removing the rest of the image. The stain is then fitted to a rectangle and the major axis is measured. Then the impact angle was calculated from the width and length of the stain. However, this had an average error rate that was only marginally acceptable and the manual process produced better results [23]. This method also relied on using a marker to identify the length of scale of the stain. This means the implementation cannot be used on similar images without this marker which is the majority of patterns photographed.

OpenCV has a simple blob detection algorithm which could be modified for segmenting blood stains [5]. The image is converted to binary by thresholding it at a range of values then the 'blobs' are extracted using the Suzki border algorithm [39]. The centers, locations, and radii are then computed and returned. Running the simple blood detection with the default settings works fairly well on images of bloodstain patterns. However, it didn't work well for elongated stains or for small stains and some stains are missed entirely. These limitations could be overcome with added steps around the main algorithm.

Bloodstain pattern analysts have used other programs to assist in identifying the origin of the bloodstain pattern. This is done with the string or tangent method. This is where analysts lay strings over the pattern in the direction that the blood droplet originated as determined by the shape of the resulting bloodstain of the bloodstain. Then the origin is where the strings converge. [20]. There are several computer programs to help with this. Most commonly BackTrack and HemoSpat are used [30]. HemoSpat has semi-automatic stain selection. The analyst can adjust the result and identifies each stain. HemoSpat then calculates the source of the blood [15]. BackTrack has a similar idea and performs as good or better results than manual analysis [17, 7]. There are also other methods proposed to automate this process such as in [22].

This idea is similar to the proportion of intersecting major axes feature looked at by Ravishka [3]. So the string method could be automated if all the stains are automatically found.

A robot automatic system for bloodstain pattern analysis has been proposed to solve this problem. With the robot using the limited colour spectrum of blood to identify and segment the blood stains, photograph them and calculate metrics [1].

While there has been interest in the automation of bloodstain pattern analysis no methods have been complete enough to be used in research because they have been tailored to the images used in the study. Resulting in inflexible solutions that are difficult to implement elsewhere. Improving these methods would create a novel generalized approach that could be used which the specialized features required for bloodstain pattern analysis.

2.3 Threshold Algorithms

Thresholding an image is often done to separate the foreground of an image with the background. As the laboratory images taken at ESR only have blood stains on them with a light plain background this is a viable method of segmenting the stains. There are several types of thresholding that could work for these images.

Otsu thresholding is a way of automatically selecting the threshold value when segmenting an image when the histogram of the intensity of the image is bi-modal. The blood pattern images are on a white, grey or light brown background and the blood stains are a darker red colour with nothing

else in the image which means the histogram becomes bi-modal ideal of Otsu thresholding [31].

The triangle method of automatic thresholding was used by Ravishka M. Arthur [3] for finding the bloodstains. It draws a line between the highest point of the intensity histogram and the farthest end of the histogram. The threshold value is then the maximum distance between the line and the histogram [41]. This is good for images with a skewed intensity histogram. Which was ideal for her images.

These methods only work well on some images. Therefore a type of adaptive thresholding may be beneficial as the thresholding more robust. This is as it thresholds part of the image at each time determined by a block size. So, it works on images with variable lighting. This could improve upon the method used by Ravishka.

2.4 Machine Learning with Bloodstain Patterns

Machine learning techniques have been used to identify and get metrics from bloodstains. A genetic algorithm described in [40] has been attempted. Which turned the problem into an optimization problem giving scientific credibility to the field but needs further work.

Machine learning was used in the measuring the gamma angle of a bloodstain. An active shape model was developed with multiple pictures of a bloodstain impacting on a surface at different angles [21]. Then this model was superimposed onto other stains to determine their impact angle. The bloodstain was separated from the background with a flood fill operation and Otsu thresholding similar to the method proposed by K. Boonkhong et al [23]. This was to improve the accuracy of measuring the impact angle of the stain. However, analysts do not see any benefit from this method.

A neural network has been used to determine a bloodstain pattern from features about individual bloodstains. The features from the bloodstains were extracted with MatLab's 'regionprops' and moment invariants after the edge of a bloodstain was detected with Sobel edge detector and the noise was removed with morphology operations from an image with a singular bloodstain in it. Then this information was passed into a function fitting neural network and a cascade forward neural network for the purpose of pattern recognition. [37]

In recent years convolutional neural networks have been used to classify between different images. A convolutional neural network could classify between the different types of bloodstain spatter patterns. One such neural network that could be used is ResNet [16].

As the dataset for bloodstain spatter images is small transfer learning could greatly increase the accuracy of the model and reduce overfitting [36]. Transfer learning greatly reduces the number of parameters in a model so less data is required to train the model. The pre-trained model can be used as a fixed feature extractor by removing the final fully connected layer and training a linear classifier. This has the advantage of being about to produce a model that doesn't overfit the data as it is has been trained on other images for the initial weights. So, it has promise with the classification of bloodstain patterns.

3 Solution

Automated bloodstain spatter pattern analysis breaks down into two problems. Automated bloodstain pattern metric computation for analysts to quickly look at different characteristics of bloodstain patterns and, automated bloodstain pattern classification. The first of which is done with traditional image processing techniques. The second is attempted with a neural network.

Automated bloodstain pattern analysis will use high-resolution images of the patterns. The images used for this project were created and provided by the Institute of Environmental Science

and Research's (ESR) bloodstain pattern analysis research group. The patterns were laboratory generated with a wide range of variables including distance away from the paper catching the blood and the amount of blood used. This results in a realistic range of patterns created, from bloodstain patterns with one bloodstain to bloodstain patterns with thousands of bloodstains. The images are of the pattern types cast off, impact and expirated. All images have been taken at a high resolution with multiple images being taken then stitched together. Therefore, images range in size from 2MB JPG files to 39.5MB uncompressed TIF files. For consistency of compressed and non-compressed file types the range of sizes in pixels for each pattern type can be seen in table 1. In order to stitch these images together, pins or stickers have been placed in the images. Therefore in addition to the bloodstains, most images have pins in the image as well as rulers around the edge to show the scale.

Table 1: Range of sizes of images in pixels

Pattern Type	smallest dimensions	largest dimensions
cast off	6878x6022	12325x2978
expirated	1478x1441	7172x6506
impact	5389x3561	13312x4644

3.1 Automated bloodstain pattern metrics

The proposed process of calculating metrics from the images is split into two parts, bloodstain segmentation, and metric computation. The metrics computed for each bloodstain are the area in pixels and area in millimeters, regularity of element margin, the circularity of the bloodstain, bloodstain direction, angle and gamma, width and height of the fitted ellipse, and intensity. The metrics computed for the pattern where the linearity, convergence, and distribution of the bloodstains.

The proposed solution takes an image is input and it produces a list of boundary points, a comma separated file containing one line for each stain and columns for each metric similar to table 2, 3, a comma separated file containing the pattern metrics similar to 4, three graphs, one for convergence seen in figure 11, linearity seen in figure 10a and one for the distribution seen in figure 10b. This workflow is represented in figure 1.

The decision was made to implement this application in Python and Open CV. This is as Python has many modules that can be imported for easy implementation of image segmentation and analysis including numpy, matplotlib, and Open CV. Open CV was used as it gave all the functionality for the computer vision algorithms that would be needed for the application. Also, it is fast as Open CV's python library is a wrapper around the C function calls. Also, neural nets are often written in Python. This meant that the same language could be used for bloodstain segmentation and pattern classification. Numpy and matplotlib were used because they are very common in python data science development. They add significant functionality in the forms of array manipulation and graph drawing that speed up the development of the application.

3.1.1 Bloodstain Segmentation

To segment the bloodstains it assumed that the image had a lighter background to the bloodstains and no other objects are in the image apart from circular stickers used for stitching. This is a reasonable assumption for research purposes. Therefore the rulers in the images needed to be cropped out.

To solved this problem an automatic cropping routine was developed. It automatically cropped the image by identifying the rulers, then adding an offset and cropping the image at those values.

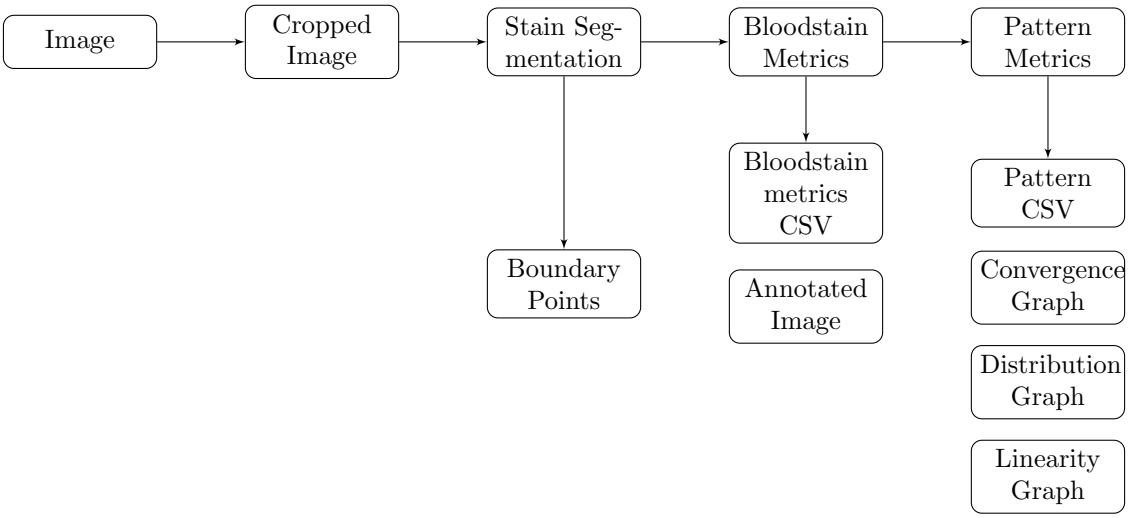


Figure 1: Representation of the inputs and outputs and how they are produced

The rulers were identified by first blurring with a Gaussian blur and converting it to grey-scale. Then canny edge detection was used to find the edges in the image [6]. Then with this result lines were found with the hough line transform [13]. Which found the dashes on the ruler. As none of the rest of the image would have as many lines starting at a similar position and close together the ruler was defined by this. These sections were found by finding the maximum number of lines that started at the same x or y value for the left, right, top and bottom of the image. Which assumes there are four rulers, one on each edge of the image. So, this procedure could only be done automatically if there was a ruler on each side of the pattern. Otherwise cropping would be done manually.

Specifically cropping was achieved by iterating over the dashes found with hough line transform and deciding if the x value is in the left half or right half of the image, and then deciding if the y value is in the top half or bottom half of the image. Then taking this into account the number of other lines that started at that same x or y value were counted. Which resulted in four lists, one for each edge of the image, with counts for x or y values. For each list, the maximum count is found and the corresponding x or y value plus an offset for is used to crop the image.

The novel approach to find the bloodstains in the image as follows. Firstly the image was converted to grey scale. Then the image was converted to binary with an adaptive threshold, seen in figure 2. Adaptive thresholding was based on the Gaussian mean value of a block in the image then a constant offset parameter. Resulting in thresholding small areas of the image at a time reducing the effect of shadows and noise [12].

An opening morphology operation was used to reduce noise and remove some spindles from the bloodstains that are not significant. This results in an image where white is the stains and black is the background. However, some of these bloodstains may actually be pins or stickers on the image that have been placed for image stitching. Stickers which are circles will then be removed by identifying circles in the image of the correct size using the Hough circle transform [4]. Finally, each stain is found by finding the contours of this image. This was done with the Suki border algorithm [39]. Which produces a polygon outlining each bloodstain which can be analyzed.

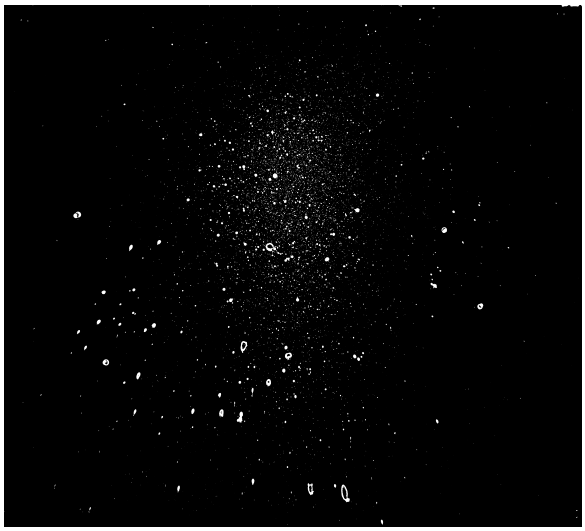


Figure 2: Binary image of a bloodstain spatter pattern as a result from adaptive thresholding

3.1.2 Metric Computation

Many metrics were calculated with firstly fitting an ellipse to the bloodstain polygon using the least squares method [14]. However, the tail of the bloodstain must be removed before doing this. This is achieved by identifying the concave points along the bloodstain boundary. Then starting from the furthest point away from the centroid the algorithm steps round the stain in both directions measuring the distance between the two points. When the ratio of this distance to the maximum width or height of the bloodstain is great than 0.3 and one of these points is a concave point then it is the end of the tail. All points between these two points are removed so the tail is removed. Then the ellipse is fitted to the resulting polygon without the tail. Six metrics were computed per bloodstain. Directly from the ellipse, the circularity was measured as a ratio of the width and height of the ellipse.

The area of the ellipse was simply a count of all the pixels in the bloodstain boundary. The intensity is the normalized average grey scale value for the bloodstain. The angle of the ellipse was measured clockwise from vertical from the center of the bloodstain. This was between the range of 0 and 179 degrees. The gamma angle is measured from the bottom of the stain in the range 0 to 359 degrees [20]. Therefore the angle given by the ellipse could be used with the direction of the bloodstain to calculate the gamma angle. It was computed by taking the angle of the ellipse and adding 180 degrees if it was going to the left.

The direction was calculated based on the angle which identified if the bloodstain was going up or down. The horizontal direction was determined which side of the stain was more rough [20]. The rougher side was calculated by splitting the bloodstain in half vertically and comparing the bloodstain area to convex hull area ratio. If the difference between the two ratios were less than 0.005 then the direction cannot be determined. Otherwise, the horizontal direction was the side with the lowest ratio. Combining this with the vertical direction from the ellipse angle gave the direction.

Similarly, the ratio of the area of the convex hull to the area of the bloodstain was a measure of the regularity of the bloodstain.

Three pattern whole metrics were calculated, linearity, convergence, and distribution of elements.

The linearity was calculated from getting all the bloodstain centroids and fitting a two-degree curve to the points with numpy's 'polyfit' [11]. This was then graphed with matplotlib.

The convergence was calculated by finding the intercepts of the paths that the bloodstains took to get to their location. The major axis of the ellipse was calculated and this line was drawn back to the edge of the image. Then a sliding window algorithm over these line segments found all the intersections. A density matrix was calculated with the Gaussian kernel density estimator described in [26] with 25 bins and then plotted which produced a nice result. Finally, a box was drawn around 60% of the intersections.

The distribution of elements metric was calculated as simply the total area of the stains divided by the total area of the pattern's convex hull.

3.1.3 User Interface

A user interface was created for usability. This included features of picking an image, choosing which annotations to display and pattern metrics to compute and, exporting the results. There are two sections of the tool. A tab view with two tables, one for the bloodstain element metric data to be displayed and one table for the pattern data to be displayed. The other view is the annotated image which can be zoomed in and panned across. One of the major features was to be able to click on an item in the list of data points and then be shown where it is on the image. This provides a link between the data and the annotated image. This is seen in figure 3 with the highlighted row and blue square on the image.

The annotated application also provided batch processing so that an entire folder can be left to process and output to another folder specified by the user with no user interaction. This will be useful for when a researcher is gathering data from many patterns.

This graphical user interface (GUI) was created with PyQt [27]. Other user interface frameworks that are more trendy and recently implemented were considered, such as Kivy. But PyQt was chosen as it is easier to implement than Tkinter and Kivy, as it is well documented and has a graphical designer, and it similarly structured to GUI frameworks I have used in the past. Finally, it is commonly used in industry for python projects.

3.1.4 Software Design

This project is structured around the core computation module that calculates model and interfaces to this module. A diagram of this is seen in figure 4. The core computation module is the stain segmentation and metric calculation. After computation, it stores the outputs of the stain segmentation, inside the pattern object which is made up of many bloodstain objects that have been segmented and the pattern metrics. This is similar to the model, view, controller architecture with the view in the CLI and GUI, the controller being the core computation module and the model being the pattern and bloodstain objects resulting from this computation.

This application follows the single responsibility principle as the pattern and stain data are separated from the core computation module which is separate from the different interfaces, CLI and GUI. The tell don't ask principle is upheld with the bloodstain object holding the calculations to calculate the bloodstain metrics and the pattern object holding the calculations for the pattern metrics. Which results in it is easy to add new metrics onto this application as another method can be easily created in either the bloodstain or pattern class which will not affect any existing code. Which is important if more metrics are added in the future.

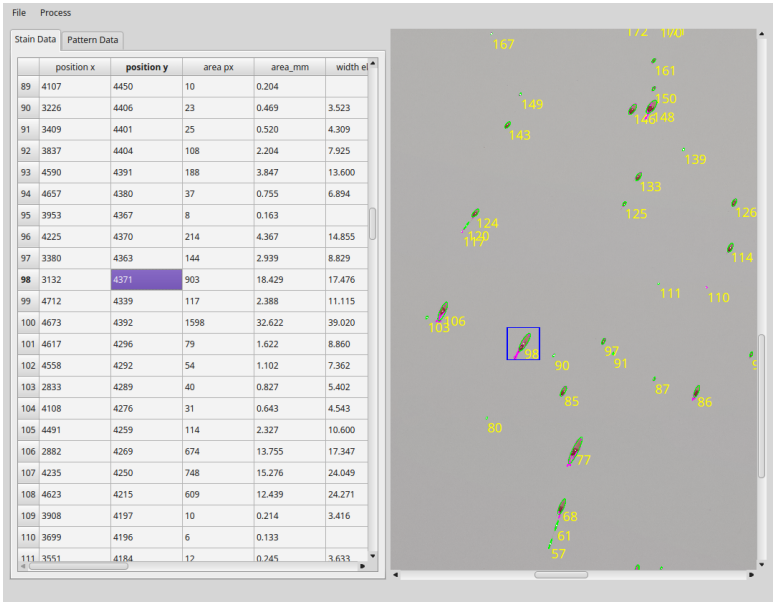


Figure 3: Graphical user interface of the program showing the table view of some bloodstain metrics with an annotated and zoomed in pattern image that has been segmented.

3.2 Pattern Classifier

To make the analysis of bloodstain spatter patterns objective an accurate a neural network could classify between the different types. A convolutional neural network type approach on the images provided by ESR was considered. But as the images are so large because of the high resolution needed to capture all the small bloodstains this approach would require large GPU memory for training. So, it is not possible on the equipment available and CUDA runs out of memory for even a small network, as all training was done on an NVIDIA 1080 GTX graphics card.

Several approaches to solving this problem were considered including cropping the images to a small size, using a sliding window approach of reading the image in and, using downsampled images. Downsampling the images was quickly ruled out because as when the images were downsampled they lost all the important information. This is because the bloodstains are small and can be sparse only covering 1% of some images. The sliding window approach had similar issues as it may not see and bloodstain in the sections or be able to get an idea of the entire pattern shape.

The images could not be randomly cropped to a small size because as some bloodstain spatter patterns are very sparse then only one or two stains would be seen in the resulting image and not the core pattern. This would mean that the neural network would not be able to build a good model. Instead, the novel approach of cropping the images to a fixed size around the area with the most bloodstains was used. This was calculated as the highest density region of the center points of each bloodstain identified with the segmentation method discussed above.

The proposed approach formed a training dataset comprised of 136 cast-off bloodstain patterns, 100 expirated patterns, and 388 impact patterns and, a validation set comprised of 27 cast-off bloodstain patterns, 21 expirated patterns and 42 impact patterns, and a test set comprised of 6 cast-off bloodstain patterns, 7 expirated patterns and 9 impact patterns. All of which have been cropped to 1000 by 1000 pixels around the highest density of bloodstains in the pattern. This

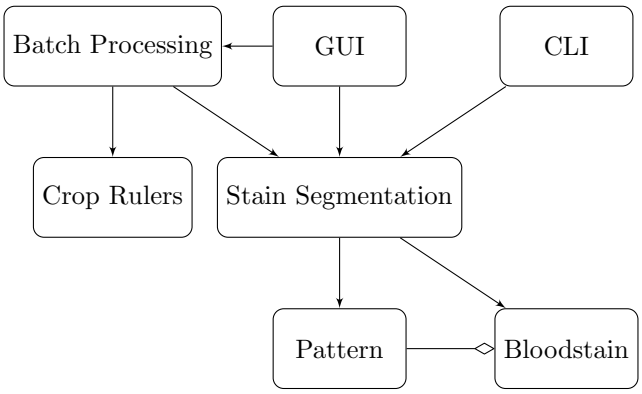


Figure 4: A high level overview of the sections of the application and how they interact

resulted in the highest change that the image contained the most important part of the pattern. The 1000 by 1000 images were transformed before being trained on by randomly flipping the images to ensure the neural network doesn't learn on the orientation of the images. The images were also shrunk to 244 by 244 so that batch normalization could occur to gain better results.

This training data set was used to train a deep residual convolutional neural network with 18 layers commonly called ResNet [16]. This state of the art neural network architecture was chosen because it has been pre-trained on image net data so transfer learning can be applied. Transfer learning was advantageous as it means that most of the parameters have already been learned and only the final connected layer needs to be trained. This means that the neural network can be effectively trained on a relatively small dataset. It also reduces overfitting as most parameters were learned on different images. A shallower neural network was used for similar reasons as it reduces the number of parameters also reducing overfitting.

A convolutional neural network performs unsupervised learning to learn the features of an image with a different feature being learned on each layer. It does this by applying convolutions on the image and adding weights to each feature. These weights are passed into a fully connected layer to compute a classification. ResNet is a residual network which solves the vanishing gradient problem and reduces the number of parameters that optimization is performed on. This is achieved by adding a path between the layer module inputs and layer module outputs implying the identity function. This means that later layers will not have to learn the identity function but the features on top of the identity function. These features are known as the residual and the module is called the residual module [16]. Because of this, it is one of the best architectures for transfer learning with a small data set as it is deep with fewer parameters.

Pytorch was the framework used to train this model. It was chosen as there was support for it in the wider research group and it provided enough granularity to adjust the network if need be and enabled the use of pre-trained models.

Another proposed approach was to train on the metric and metadata calculated from segmenting the bloodstains. When the bloodstains are segmented there is a polygon drawn around the edge of the bloodstain. The points of all the polygons could be visualized as a two-dimensional point cloud. Therefore the point net neural network was used to train a neural network on these points [32]. The small polygons that had an area less than 10 pixels were discarded so there was less noise in the points. The base number of features used in the neural network was reduced from 64 to 16 as the data set is small to avoid overfitting the data and because the data set is small. To condense the

points into a unit space the points were first normalized before being trained on.

Point net 2 is a deep convolutional neural network. It improves upon point net [33] by capturing the local structure of the points. It does this by grouping points together in neighbourhoods in a hierarchical fashion. Then using point net as a local feature detector which is effective architecture to process un-ordered sets of point for feature extraction. Therefore it will be able to group points to do with a particular bloodstain together and learn on this.

4 Software Engineering Processes

A Trello board has been created to keep track of the features ESR wants to be implemented and to continue to work on the long-term goal by breaking it down into stages. With use of the Trello board a kanban type approach has been followed with tasks being worked on one at a time from a prioritized backlog. Some tasks have needed to be validated by an analyst and so ESR has tested them. This lead to incremental improvements in the solution to the problem. Team-based software engineering processes are not that applicable to this project due to it being an individual project.

Communication is important in agile processes. This has been managed with regular meetings. Once a week a meeting is held where ESR can see the latest updates. They will raise any questions or ideas including new features to implement. The every changing requirements of the client, ESR, have been managed by discussing what is the most important to implement in this meeting. The backlog is then updated.

This meeting is also important to exchange domain knowledge. As ESR want to continue the project next year it is important to inform them of the technologies used and explain some of the techniques used for segmenting the bloodstains. Conversely, in order for the features to be implemented the domain knowledge about bloodstain pattern analysis must be conveyed. This has included a tour of ESR's laboratory to demonstrate how the bloodstain patterns are made. Generally, there were many questions from both parties.

This was often assisted with a demo of the project which often leads to finding bugs or more features being requested. If the feature was with the user interface then user stories were written and added for clarity on why the feature was needed which helped in prioritizing the backlog.

The only hold up in development was that there wasn't an image data set at the beginning of the year. ESR was actively making bloodstain spatter patterns to photograph and form a data set for the first half of the year. This meant that the work on creating a solution to classify bloodstain pattern types with these images was not possible. So, more time was spent on other aspects of the project.

Good code practice has been followed in this project. The version control system git has been used. There was a main branch with the copy that ESR has access to so they can see the state of the project. Then for each feature, a branch was made then merged into master when it was finished. Bugs were fixed on master as there is only one person developing on the project. The code follows the PEP8 style guide for Python for constancy and an emphasis on readability has been placed on the code as I am aware that forensic scientists may want to modify it in the future.

Testing was performed by both myself and a domain expert. The domain expert has access to the original patterns so they could check that the metrics produced were accurate. The segmentation and metric computation were tested on all images provided by ESR. This caught any edge cases found as a result of the segmentation. It would be difficult to test that the solution was correct by an automated test, therefore, all the testing was done manually. Also as the solution evolved over time and improved unit tests were not feasible for this project. A manual testing document was not needed as this project was effectively tested by running the main segmentation script as all the logic

was in this script. As the user interface is not complicated user interface tests were not needed. As the user interface and options on the main script add to the functionality and the ideal results are identified then automated testing would be feasible to use.

5 Results

5.1 Bloodstain Segmentation

The segmentation of bloodstains was successful. All bloodstains on the image were identified, no matter the size of the bloodstain or lighting set up. It will work on images that do not have a white background but the background must be sufficiently lighter than the bloodstains. An example of the successful segmentation of part of an expirated image can be seen in figure 5 and figure 6. However, there are some false positives in the images. This is due to the assumption that only bloodstains are in the image. But the images actually contain markers in the form of drawing pins or stickers used to stitch the images together. While the removal of the stickers in a circle shape was successful other markers were not removed as seen in figure 7b and figure 7a. If the images do not have their rulers cropped off then there will be false positives of the ruler markings which is why the images needed to be cropped. There were other false positives in the form of other marks or screws in the image but these were less significant. In the future instance segmentation could be used to overcome these limitations.

This segmentation performed similarly to imageJ. However, both methods picked up a small amount of noise. Both of these methods perform better than Ravishka M. Arthur's method. As many of the stains in an impact pattern were missed in the segmentation using Ravishka's method, are picked up in this method and it works on many different bloodstain pattern images taken in different ways. This method takes the better segmentation from image J and combines it with the specialized metrics that Ravishka's method also implemented including some new metrics. Then it adds a friendly user interface.

There were issues in whether a bloodstain identified was not just noise in the image. This is because there are very small stains. However, it was decided that a bloodstain identified must be over three pixels in size to be included. There is some noise identified as stains at the edge of an image. This is because of the edge of rulers or the edge of the background is visible due to insufficient cropping. There is also some shadow at the edge of the images which contributes to this.

When two stains are close together or touching then the segmentation will only identify one bloodstain. Which can be seen in figure 8a This is due to it appearing to be one region. Also, satellite stains are still found which is not desirable as they have come off the main bloodstain. This is seen in figure 8a as the small stains around the big bloodstain that have been circled. However, these bloodstains are difficult to filter out because some main stains are smaller than satellite bloodstains and bloodstains cluster together in some patterns.

Ellipses are not fitted well to some stains. This includes dripping stains such in figure 8b and stains with multiple large spindles. It tends to not include the lighter area where the stain has dripped from and includes the end of the drip as well as the trail back up the image. However, the removal of tails worked well for stains with proper tails and stains without tails. This is seen in figure 8c, as the green ellipse not been fitted to all the pink contour instead it has been fitted to the main area and not the tail.

The automatic cropping of images removed the rulers well but only if there were four rulers over the pattern on each edge, and if they were mostly horizontal or vertical. Otherwise, the image would have to be cropped manually again to remove the rest of the background and ruler. The offset amount was adjusted for each type of image as rulers have different thicknesses and a larger offset

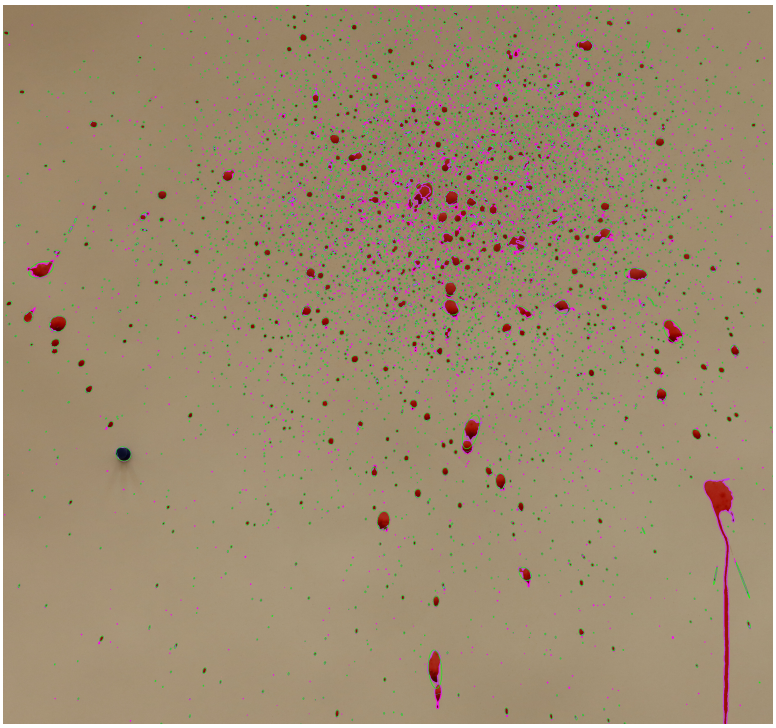


Figure 5: The main part of an expirated bloodstain pattern after segmentation. With ellipses fitted in green and the outline shown in pink.

may be necessary if the ruler is on an angle. Consequently, this is not a general method for cropping the images. This could be improved upon to be more robust.

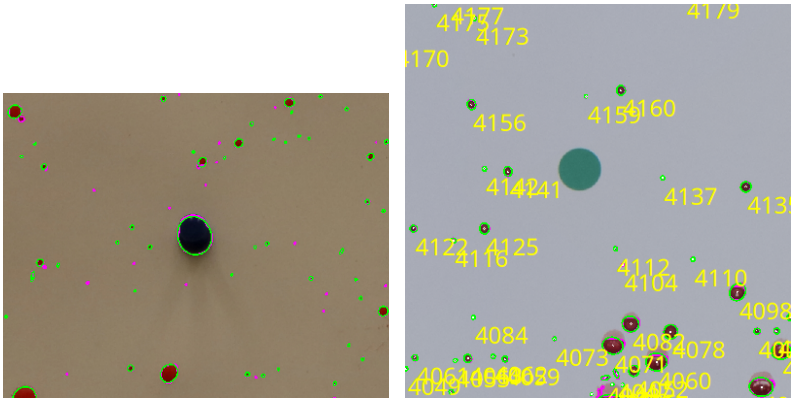
5.2 Metric Computation

The initial segmentation including fitting the ellipses was very efficient. It can take from around two seconds to up to around nine seconds to compute depending on the size of the image and the number of stains found. Finding all the element metrics was a linear operation as each stain had some operations done on it and none of the metrics had a large complexity. As a result as the number of stains increased the amount of time it took to generate the metrics increased. This can be seen in figure 9.

5.2.1 Bloodstain Metrics

The bloodstain metrics were computed accurately. An example output of the stain metrics can be seen in table 2 and 3. These were calculated accurately as tested by ESR. Some stains do not have enough points around their contour to fit an ellipse to it. This results in empty values for the width and height of the ellipse as well as the direction, angle, and gamma which are calculated from the ellipse. The direction is sometimes unknown because on some stains with even roughness all the way around so, an analyst can not determine the direction. It was found that most bloodstains that the very circular, the direction is difficult to tell. As well as this the angle, gamma, and therefore

Figure 6: The main part of a castoff bloodstain pattern after segmentation. With ellipses fitted in green and the outline shown in pink with the stain is in yellow.



(a) A pin being segmented (b) A sticker not being segmented

Figure 7: 7a). A pin being segmented like a bloodstain as a false positive. 7b). A sticker being ignored and not segmented because of the hough circle transform

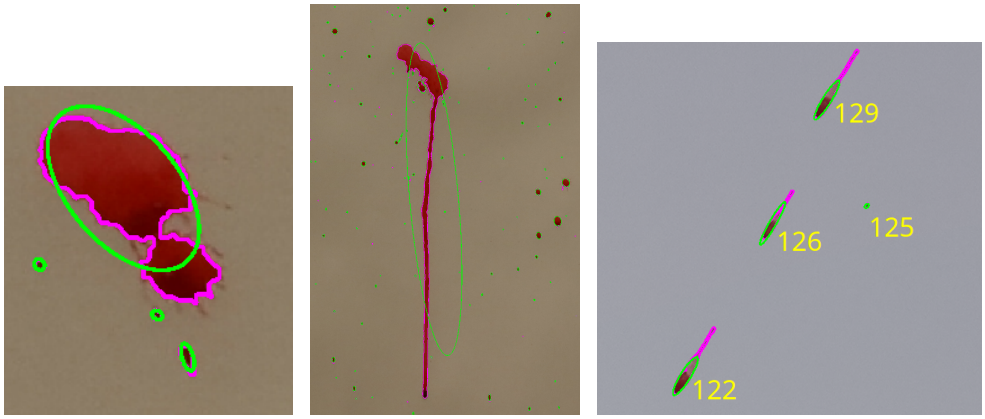
direction is shown in table 3 becomes more unreliable when the stain becomes more circular because the difference between the width and height becomes small and it tends to not contain spindles which contribute to the roughness.

5.2.2 Pattern Metrics

Numerical data was exported in from the pattern metrics for analysis as well as producing some graphs. The numerical data from a cast off pattern is shown in table 4. This data can also be seen in the graphs outputted but is useful in this format for comparison between patterns.

The linearity output was a graph with all the stain locations plotted then a polynomial was fitted to the points. The output of a cast-off pattern for this metrics is seen in figure 10a. It can be seen that the R^2 value is presented with a line that fits well to the stain locations. This R^2 value and the polynomial function were outputted for comparison.

The distribution outputted the ratio of the area that the stains took up to the area that the convex hull of the stains had. This was also shown as in figure 10b to show were the convex hull is. This metric lost a bit of its meaning because of false positives. The markers in the form of stickers or pins were are regular intervals around the pattern and noise was identified at the edge of the



(a) Ellipse fitted to merged bloodstains (b) Poorly fitted ellipse (c) Ellipse fitted well

Figure 8: 8a). One ellipse being fitted to two stains that have merged together. 8b). An ellipse being fitting poorly as the stain has dripped. 8c). An ellipse being fitted well with the tail removed

image and classed as bloodstains. So, these were included in the convex hull so it made the convex hull much bigger than it should have been. In figure 10b the far right points are probably noise, removing them would make a better convex hull.

The convergence took much longer to calculate than all other metrics. This is due to calculating the distribution of the intersections to find the point with the highest density. This was done with the kernel density algorithm using Gaussian kernels. There as there can be many stains then there can be many intersections along the paths the blood droplets took to reach their final destination. In some cases reaching over 10000 intersections. So calculating the kernel density estimation algorithm becomes very slow.

This can be mitigated with reducing the binning of the intersection points however it creates a less detailed plot. As seen when comparing figure 11b to figure 12. It takes around 2 seconds to compute convergence with binning of 100 for 1975 intersections and 12.2 seconds for 1975 intersections with binning of 400. Showing that there is not a linear relationship between the binning amount and the amount of time it takes. Therefore increasing the binning amount drastically increases the computation time. A binning amount of 25 was decided upon because with 750396 intersections it takes around 15 minutes to compute but lower than this the density graph becomes misleading and with too low resolution.

The convergence outputted the box that contained 60% of the stains and the convergence point which is the point of highest density. This was also graphically displayed as seen in figure 11. Each * on the upper plot is an intersection point with the image behind. Each blue line is a path that the bloodstain traveled to get to its resting place which can be toggled off.

The convergence becomes less reliable when a high percentage of bloodstains have an equal amount of roughness on each side of the stain or are very circular. This is because the convergence relies on the calculation of the direction that the stain is traveling and this becomes unreliable when the bloodstain is very circular or has an equal roughness around the stain. To help with this only stains with a width to length ratio below 0.86 were used for convergence, this represents all stains with impact angle up to 60 degrees. At angles greater than this get difficult to measure as they less

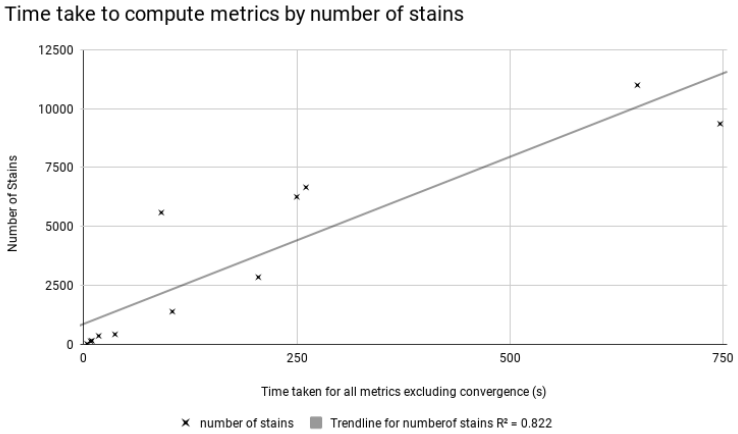


Figure 9: Time taken in seconds for all the metrics apart from convergence to be computed and the number of stains identified

Table 2: Bloodstain metrics values read from the ellipse and contour as part of a cast-off pattern

id	position x	position y	area px	area_mm	width ellipse	height ellipse
0	5506	2711	4292	87.592	71.446	75.241
1	647	2457	639	13.041	20.504	40.016
2	1308	2366	224	4.571	12.259	24.746
3	1100	2345	52	1.071	6.432	12.467
4	1608	2312	379	7.735	17.742	27.997
5	2782	2209	689	14.071	25.548	35.284
6	2670	2195	21	0.439	5.622	6.388
7	2517	2206	925	18.888	28.78	41.798
8	1526	2157	143	2.929	10.252	18.712
9	343	2140	5	0.112		
10	328	2139	24	0.49	3.118	12.865

elliptical it improves the accuracy of the convergence.

5.3 Usability

Initially, this project was only a command line interface but to make the solution usable for a blood spatter pattern analyst it needed a graphical user interface (GUI). This resulted in GUI with a table and image views seen in figure 3. This interface makes it simple to analyse an image, there are two steps. First, the user loads a file with file → load then it is shown on the right in the image view. Then the user analyses the image with process → 'Segment Image'. A dialog box shows up and allows the user to pick the annotations and which pattern metrics to compute. These were included because the convergence takes a long time to process so a user may not want to produce this metric. The choice of annotations was included because the id annotations can overlap with each other so the user can only see a yellow blob instead of the bloodstains. A progress bar appears while the pattern is segmenting. When it is finished segmenting the stain data and pattern data tables will

Table 3: Bloodstain metric values from part of a cast-off pattern calculated from the fitted ellipse and contour

id	angle	gamma	direction	solidity	circularity	intensity
0	152.014	152.014	?	0.986	0.95	0.158
1	85.018	265.018	('left', 'down')	0.962	0.512	0.183
2	86.086	266.086	('left', 'down')	0.972	0.495	0.238
3	85.844	265.844	('left', 'down')	0.972	0.516	0.345
4	83.246	83.246	('right', 'up')	0.976	0.634	0.216
5	87.778	87.778	?	0.966	0.724	0.178
6	72.121	72.121	('right', 'up')	0.977	0.88	0.431
7	89.691	269.691	('left', 'down')	0.97	0.689	0.177
8	86.46	86.46	('right', 'up')	0.953	0.548	0.268
9			?	1		0.377
10	81.211	81.211	('right', 'up')	0.873	0.242	0.415

Table 4: Pattern metrics output values from a cast-off pattern

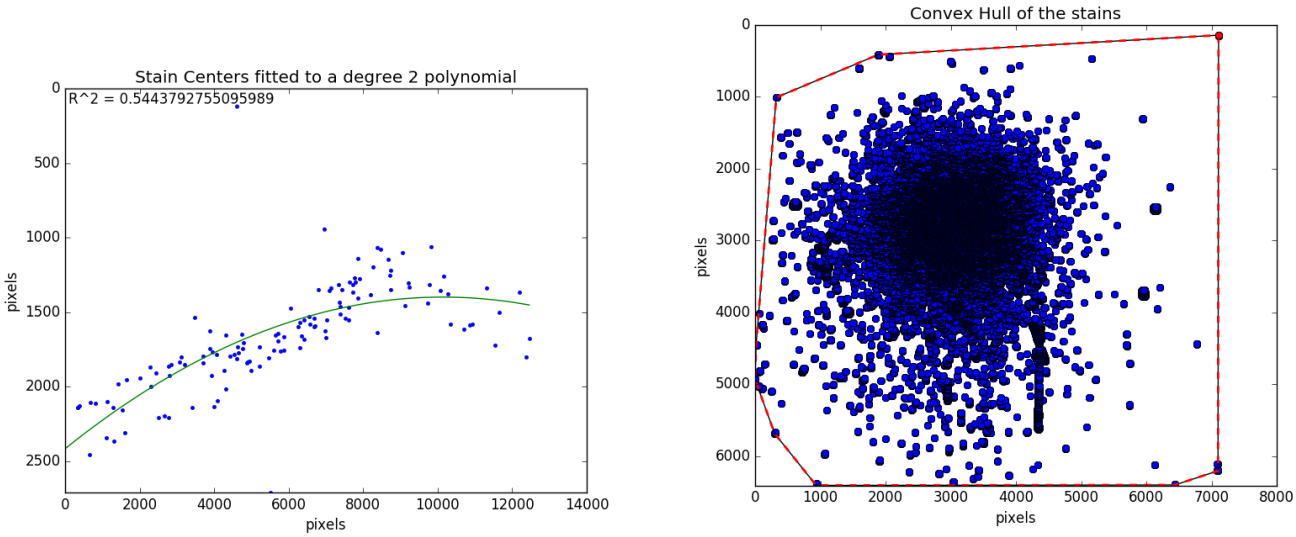
Linearity - Poly-line fit	$9.887e-06x^2 - 0.2006x + 2417$
R^2	0.5444
Distribution - Ratio stain number to convex hull area	6.933e-06
Ratio stain area to convex hull area	4.154e-03
Convergence - point of highest density	(4148.6, 1881.3)
box of %60 of intersections	lower left (x,y) : (2894.5,1321.0)
	Width : 2090.2
	Height : 1611.0

populate

While this GUI is usable it could be improved to be more user-friendly. For example, it could be improved with a more accurate progress bar, adjusting the size of the image area, integrated cropping, interactively removing pins or other marks on the image that they think are stains, turning on and off annotations. Most importantly the annotations could maintain the same thickness when zooming in an out on the image view. This will allow the user to see the bloodstains as well as the annotations and not have the annotations covering all the image. Solving the yellow blob of bloodstain ids in an expirated pattern where all the ids overlap.

This interface is easier to use than image J and has one feature that makes it substantially better. This is the ability to click on a bloodstain stain entry in the table and be shown which stain it corresponds to. In image J the ids for the segmented bloodstains are shown in a low resolution black and white image where the number is located at the point where the bloodstain is which is also hard to read [19]. This makes the workflow for looking at bloodstain metrics then seeing why that data is like that difficult as the analyst has to find the number on the image J id image then map it to the real image of the bloodstain in the large pattern which is difficult when there are many stains close together. This feature streamlines this process because all they have to do is click on the data entry they want to see the bloodstain for.

Another ease of use feature is batch processing. While batch processing is possible with ImageJ with scripting [18], batch processing is built-in in this solution. An analyst can easily select which folder to batch process and set it to run. It will run each segmentation in a new subprocess so not



(a) Linearity with fitted polynomial line

(b) Convex Hull for stain distribution

Figure 10: Linearity and distribution graphs

use all the memory and the computer can still be used while processing. This will be useful for researchers with lots of images.

5.4 Pattern Classifier

5.4.1 Transfer Learning

Transfer learning from resnet with 18 layers on the cropped images around the highest density region was quite successful. It had a validation accuracy of 92.2% with a loss of 0.2326. It took three minutes to train on an NVIDIA 1080 GTX graphics card. There was a 100% accuracy on cast-off patterns and an 80.9% accuracy on expirated patterns and a 92% accuracy on the impact patterns. For the test set, there was an 83.3% accuracy on cast-off patterns, 71% accuracy on impact patterns and 71.4% accuracy on expirated patterns with a total accuracy of 77%. However, the test set was very small with only six cast-off patterns, nine impact patterns, and seven expirated patterns so in reality there was only one incorrect cast off classification, two incorrect impact classifications and two incorrect expirated patterns.

As expirated patterns and impact patterns are very similar it was hypothesized that there would be confusion between the two patterns, however, this doesn't seem to be the case with the patterns being labeled as other patterns evenly between the patterns. The cropping also removed the scale of the images but as most images were taken at around the same scale this didn't affect the classification either.

When the prediction is incorrect, cast off was predicted the most with 66.6% of the time or eight incorrect predictions out of 12. Expirated is the least likely to be chosen as it is chosen 0.08% (or one out of 12 incorrect predictions) of the time. Impact is also less likely to be chosen with it chosen 25% of the time or three out of 12 incorrect predictions. From this, it can be inferred that the

larger impact data set size have not had an impact on the number of times it is predicted. It is also inferred that the reason why cast off is being predicted because only a few large stains are in this image which may be the case because there wasn't as much blood used when creating the image. So when the image is cropped then only these stains remained so it looked similar to a cast-off cropped pattern.

The quality of the results is limited to the quality of the data the neural net is trained on. As the images were cropped some important data was lost. The images in figure 13 are visually similar because they have lost the rest of the patterns, which can be seen when they are when compared with 6 and 5. Cast off patterns usually have a trail of bloodstains across the page, but as the images are cropped only have part of this trail as in figure 13b. Therefore they can look very similar to expiated and impact cropped patterns. Expirated and impact pattern both can have regions of close together bloodstains so they both look visually similar. Resulting in incorrect predictions due to losing the rest of the pattern. However, the majority of cast-off cropped bloodstain pattern images are distinctive, which has lead to good results.

To improve this result images with more of the pattern would be needed to be used. For example trying this approach with 2000 by 2000 images down sampled to 1000 by 1000 may give better results. It also may be worthwhile taking parameters from further up the layers and training a state vector machine instead of linear classifier to see if it improves.

5.4.2 PointNet++

The novel approach to classify the patterns with PointNet++ was not as successful. It seemed promising as it reached an accuracy of 88% on the validation and test sets at a loss of 0.035. However, the training accuracy also didn't increase above 87%. While this is promising the network could not determine the difference between the impact and expirated patterns and would label every impact pattern as an expirated pattern. It has 100% accuracy on the cast off patterns and 100% on impact patterns and a 0% accuracy on expirated patterns. This is because the points extracted from the impact and expirated patterns are very similar.

Experimentation with the size of the local neighbourhoods within PointNet++ may improve this result but it might be more worthwhile to incorporate other metrics into the training because the inferred location and size of the stains from the bounding points of the stain is not enough information to decide if it is an expirated or impact pattern type. This could be a different machine learning approach.

6 Conclusion

A novel approach to automatically computing bloodstain spatter pattern metrics is achieved in this project. While there is a limitation around other objects in the image being included in the segmentation as well as incorrect fitting of some ellipses, it accurately and quickly segments all stains and calculates the metrics much faster than manual analysis. It improves upon imageJ and Ravishka M. Arthurs applications. This is as it has the specialized metrics that bloodstain spatter pattern analyst will want to look at improving upon imageJ's basic metrics and improved user interface [38]. It improved on Ravishka's M. Arthur's method as it segments all the bloodstains with varied lighting including shadows and on different backgrounds [3]. The user-friendly graphical user interface will allow researchers to easily generate the metrics so that they can look at more characteristics on all the patterns they produce in the future. Making this a valuable research tool.

There is significant promise in the ability of a convolutional neural network to distinguish between and classify different patterns. Specifically, this is achieved by cropping the densest 1000 by 1000

pixel area and training on a scaled, 244 by 244 pixel version. However, more data is needed for the patterns and other pattern type data collected to solidify these claims.

More investigation into the use of the metric data for pattern classification is needed. The use of unordered point set classification using PointNet++ showed initial promise, but could not distinguish between impact and expirated patterns.

6.1 Further Work

6.1.1 Bloodstain Instance Segmentation

To extend simple segmentation of bloodstain patterns a convolutional neural net could be used. They have the advantage that it would be able to identify bloodstains on surfaces where thresholding doesn't work such as on an absorbent or patterned background. Instance segmentation an extension of a simple convolutional network would be performed to identify each bloodstain [28, 24]. It has been used on similar images with good results including segmenting nuclei of cells in a given microscopy image [35].

6.1.2 Automated Blood Stain analysis Tool

The graphical user interface could be improved upon. Most notability, the annotations could maintain the same thickness when zooming in an out on the image view. This will allow the user to see the bloodstains as well as the annotations and not have the annotations covering all the image. It will solve the problem of not being able to read the stain id's when they are overlapping with each other.

6.1.3 Pattern Classifier

The pattern classifier is not at 100% so more work is needed to increase the accuracy. This could be done with adding more transforms or modifying other parameters. As the data is quite different to what the model was trained on it might be worthwhile to use the parameters from further up the layers then train a support vector machine instead of a linear classifier. If a large GPU is used then using the 10000 by 1000 images could be trained on without downsampling resulting in higher resolution stains so that the neural can identify the minor differences better.

A machine learning approach could be done on the metric data produced from the stain segmentation code. This would take into account the entire image instead of only the cropped region and thus produce a better model. It could also provide further insight into which metrics are important.

Acknowledgments

The authors would like to thank Ravishka M. Arthur for a copy of her method to compare this method with, The Institute of Environmental Science and Research, especially Rosalyn Rough, for the photos of blood spatter patterns used in this project and domain expertise, and finally Oliver Batchelor for mentoring in neural networks and Pytorch.

References

[1] G. Acampora et al. "Towards Automatic Bloodstain Pattern Analysis through Cognitive Robots". In: *2015 IEEE International Conference on Systems, Man, and Cybernetics*. Oct. 2015, pp. 2447–2452.

[2] Ravishka M. Arthur et al. "A novel, element-based approach for the objective classification of bloodstain patterns". In: *Forensic Science International* 257 (2015), pp. 220–228. ISSN: 0379-0738. DOI: https://doi.org/10.1016/j.forsciint.2015.08.028. URL: http://www.sciencedirect.com/science/article/pii/S0379073815003722.

[3] Ravishka M. Arthur et al. "An image-processing methodology for extracting bloodstain pattern features". English. In: *Forensic Science International (Online)* 277 (Aug. 2017), pp. 122–132. URL: http://search.proquest.com.ezproxy.canterbury.ac.nz/docview/1919442265?accountid=14499.

[4] D.H. Ballard. "Generalizing the Hough transform to detect arbitrary shapes". In: *Pattern Recognition* 13.2 (1981), pp. 111–122. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(81)90009-1. URL: http://www.sciencedirect.com/science/article/pii/0031320381900091.

[5] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[6] John Canny. "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[7] A. L. Carter. "The Directional Analysis of Bloodstain Patterns Theory and Experimental Validation". In: *Canadian Society of Forensic Science Journal* 34.4 (Jan. 2001), pp. 173–189. DOI: 10.1080/00085030.2001.10757527.

[8] National Forensic Science Technology Center. *A Simplified Guide To Bloodstain Pattern Analysis*. [Accessed: 04-05-2018]. 2013. URL: http://www.forensicsciencesimplified.org/blood/index.htm.

[9] Pamela Colloff. "Blood Will Tell, Part 2: Who Killed Mickey Bryan?" In: *New York Times* (June 2018). URL: https://www.nytimes.com/interactive/2018/05/31/magazine/joe-bryan-part-2-blood-spatter-analysis-faulty-evidence.html.

[10] National Research Council Committee on Identifying the Needs of the Forensic Sciences Community. *Strengthening Forensic Science in the United States: A Path Forward*. Washington, DC: The National Academies Press, 2009. ISBN: 978-0-309-13130-8. DOI: 10.17226/12589. URL: https://www.nap.edu/catalog/12589/strengthening-forensic-science-in-the-united-states-a-path-forward.

[11] The SciPy community. [Accessed: 24-09-2018]. Aug. 2018. URL: https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html.

[12] Open CV. [Accessed: 25-09-2018]. Sept. 2018. URL: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html.

[13] Richard O. Duda and Peter E. Hart. "Use of the Hough Transformation to Detect Lines and Curves in Pictures". In: *Commun. ACM* 15.1 (Jan. 1972), pp. 11–15. ISSN: 0001-0782. DOI: 10.1145/361237.361242. URL: http://doi.acm.org/10.1145/361237.361242.

[14] Andrew Fitzgibbon and Robert Fisher. "A Buyer's Guide to Conic Fitting". In: (Feb. 1970).

[15] FORident Software. *HemoSpat*. Version 1.11.1. Feb. 19, 2018. URL: https://hemospat.com.

[16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[17] Mike Illes et al. "Use of the BackTrack TM Computer Program for Bloodstain Pattern Analysis of Stains from Downward-Moving Drops". In: *Journal of the Canadian Society of Forensic Science* 38 (Dec. 2005), pp. 213–218.

[18] *Image J - Batch Processing*. [Accessed: 09-26-2018]. June 2018. URL: https://imagej.net/Batch_Processing.

[19] *Image J User Guide 30Analyze*. [Accessed: 09-26-2018]. Oct. 2012. URL: http://imagej.net/docs/guide/146-30.html#sub:Analyze-Particles....

[20] Stuart H. James, Paul E. Kish, and T. P. Sutton. *Principles of bloodstain pattern analysis: theory and practice*. English. Boca Raton, Fla: CRC, 2005. ISBN: 0849320143.

[21] Philip Joris et al. "Calculation of bloodstain impact angles using an Active Bloodstain Shape Model". In: *Journal of Forensic Radiology and Imaging* 2.4 (2014), pp. 188–198. ISSN: 2212-4780. DOI: https://doi.org/10.1016/j.jofri.2014.09.004. URL: http://www.sciencedirect.com/science/article/pii/S2212478014001087.

[22] Philip Joris et al. "HemoVision: An automated and virtual approach to bloodstain pattern analysis". In: *Forensic Science International* 251 (2015), pp. 116–123. ISSN: 0379-0738. DOI: https://doi.org/10.1016/j.forsciint.2015.03.018. URL: http://www.sciencedirect.com/science/article/pii/S0379073815001243.

[23] Boonkhong Kittipat, Montri Karnjanadecha, and Aiyarak Pattara. "Impact angle analysis of bloodstains using a simple image processing technique". In: 32 (May 2010).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: (2012), pp. 1097–1105.

[25] Terry Laber et al. "Reliability assessment of current methods in bloodstain pattern analysis". In: *National Institute of Justice, US Department of Justice* (2014).

[26] Thomas Ledl. "Kernel density estimation: theory and application in discriminant analysis". In: *Austrian journal of statistics* 33.3 (2004), pp. 267–279.

[27] Riverbank Computing Limited. [Accessed: 25-09-2018]. 2018. URL: https://riverbankcomputing.com/software/pyqt/intro/.

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[29] mathworks. *regionprops*. [Accessed: 02-06-2018]. 2018. URL: https://www.mathworks.com/help/images/ref/regionprops.html.

[30] Samir Kumar Bandyopadhyay Nabanita Basu. "Bloodstain pattern analysis – A less explored domain". In: *International Journal of Applied Research* (Nov. 2016), pp. 200–211. ISSN: 2394-5869.

[31] N. Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (Jan. 1979), pp. 62–66. ISSN: 0018-9472. DOI: 10.1109/TSMC.1979.4310076.

[32] Charles Ruizhongtai Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *CoRR* abs/1706.02413 (2017). arXiv: 1706.02413. URL: http://arxiv.org/abs/1706.02413.

[33] Charles Ruizhongtai Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CoRR* abs/1612.00593 (2016). arXiv: 1612.00593. URL: http://arxiv.org/abs/1612.00593.

[34] William Ristenpart et al. “Quantitative analysis of high velocity bloodstain patterns”. In: *J. Abbr. 00: 000-000* (2013).

[35] Selim Sef. *[ods.ai] topcoders, 1st place solution.* Apr. 2018. URL: https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741.

[36] Ali Sharif Razavian et al. “CNN features off-the-shelf: an astounding baseline for recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 2014, pp. 806–813.

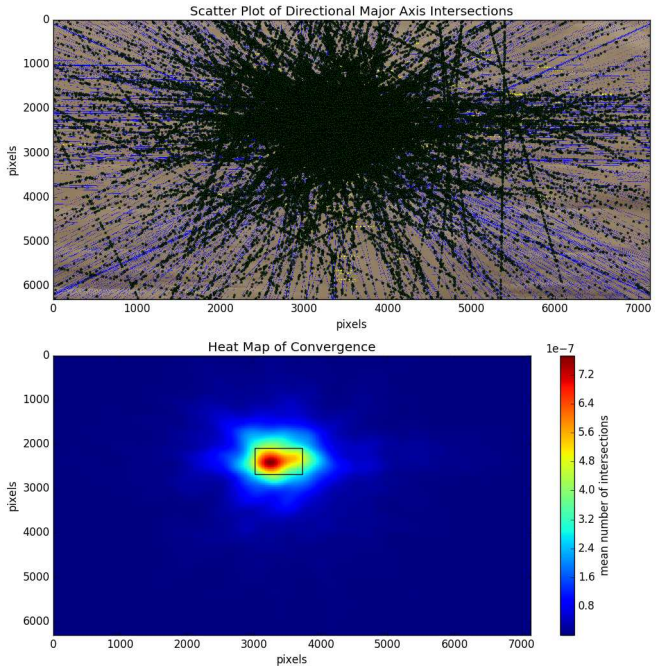
[37] N.J. Shoumy et al. “Feature extraction for neural network pattern recognition for bloodstain analysis”. In: 11 (Jan. 2016), pp. 8583–8589.

[38] Theresa Stotesbury, Mike Illes, and Andrew Vreugdenhil. “Investigation of Physical Effects of Acid Yellow 7® Enhancement on Dark and Non-Porous Surfaces in Impact Pattern Area of Origin Estimation”. In: 45 (Mar. 2012), p. 22.

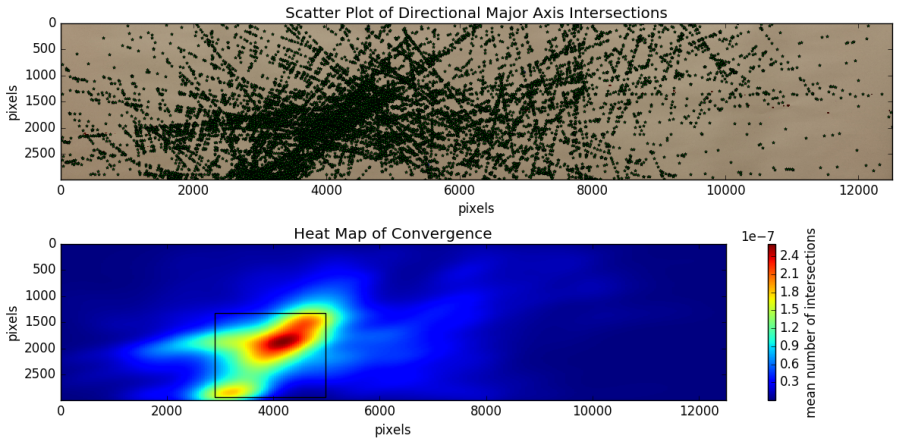
[39] Satoshi Suzuki and KeiichiA be. “Topological structural analysis of digitized binary images by border following”. In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46. ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(85)90016-7. URL: http://www.sciencedirect.com/science/article/pii/0734189X85900167.

[40] Autilia Vitiello et al. “Bloodstain pattern analysis as optimisation problem”. In: *Forensic Science International* 266 (2016), e79–e85. ISSN: 0379-0738. DOI: https://doi.org/10.1016/j.forsciint.2016.06.022. URL: http://www.sciencedirect.com/science/article/pii/S0379073816302742.

[41] G W Zack, W E Rogers, and SA Latt. “Automatic measurement of sister chromatid exchange frequency”. In: 25 (Aug. 1977), pp. 741–53.



(a) Expirated Convergence



(b) Cast off pattern

Figure 11: Convergence of an expirated pattern with density maps and scatter plots of intersections with 300 bins

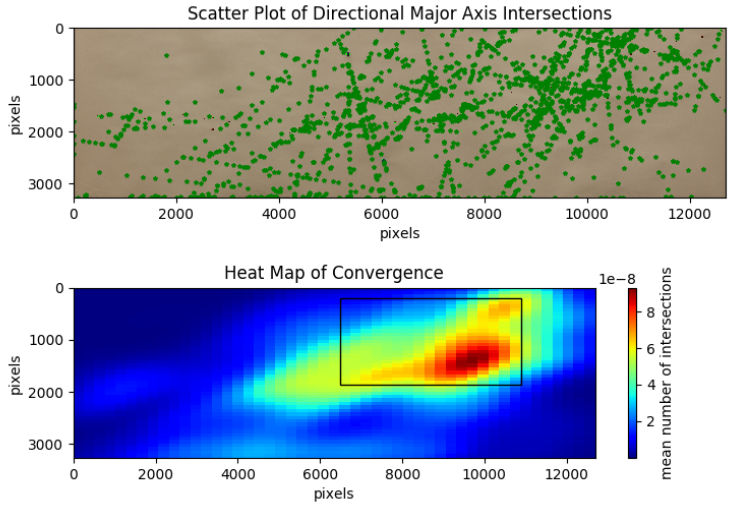


Figure 12: Convergence of an cast pattern with density maps and scatter plots of intersections with 100 bins

(a) Expirated (b) Cast off (c) Impact

Figure 13: Cropped cast-off impact and expirated patterns that look visually similar