Open-source implementation of the Congruent Matching Cells method for cartridge case identification

by

Joseph Robert Zemmels

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Statistics

Program of Study Committee:
Heike Hofmann, Co-major Professor
Susan VanderPlas
Danica Ommen
Yumou Qiu

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation/thesis. The Graduate College will ensure this dissertation/thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

DEDICATION

I would like to dedicate this thesis to my best friend and partner Emily for constantly inspiring and challenging me. I would also like to thank my friends and family for their loving guidance and support.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Heike Hofmann and Dr. Susan VanderPlas for their guidance, patience, and support throughout this research and the writing of this thesis. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Danica Ommen and Dr. Yumou Qiu.

ABSTRACT

Firearm evidence identification is the process of analyzing bullets or cartridge cases left at a crime scene to determine if they originated from a particular firearm. Statistical methods have long been developed and used to aid in such analyses. The Congruent Matching Cells (CMC) method is one such method developed at the National Institute of Standards and Technology (NIST) to quantify the similarity between two spent cartridge cases based on the markings left by the firearm barrel during the firing process. We introduce the first open-source implementation of the CMC method in the R package cmcR. The package will bolster forensic researchers' abilities to investigate, validate, and improve upon current statistical methodology in the field of forensic science.

CHAPTER 1. OVERVIEW

Matching cartridge cases based on markings left by the firearm from which they were fired is a challenging yet worthwhile pursuit. Forensic examiners have long performed this process by closely analyzing markings under high-resolution comparison microscopes. Many automated tools have been developed to supplement the work done by forensic examiners. The Congruent Matching Cells method is one such tool developed at the National Institute for Standards and Technology. This paper introduces the first open-source implementation of the Congruent Matching Cells method in the form of the R package `cmcR`. This package aims to further the field of automatic forensic pattern matching by making a "black box" method more transparent.

1.1 Introduction

A *cartridge case* is a type of firearm ammunition that contains a projectile (e.g., bullet, shots, or slug). When a firearm is discharged, the projectile stored in the cartridge case is propelled down the barrel of the firearm. In response, the rest of the cartridge case that remains inside of the firearm is forced towards the back of the barrel. The force with which the cartridge case is propelled backwards causes it to strike the back wall, known as the *breech face*, of the barrel. An example of the breech face from a 12 GAUGE, single-shot shotgun is shown in Figure 1.1a (Doyle (2019)). The hole in the center of the breech face houses the firing pin that shoots out to strike a region on the base of the cartridge case known as the *primer*. This in turn ignites the propellant within the cartridge case causing a deflagration of gases that propels the bullet forward down the barrel. Markings due to, e.g., manufacturing imperfections are ingrained on the breech face. When the cartridge case slams against the breech face, these markings can be "stamped" into either the primer of the cartridge case or the cartridge case itself. The markings left on a cartridge case from the firearm's breech face are called *breech face impressions* Heard (2008). Figure 1.1b

shows a cartridge case fired from the shotgun shown in Figure 1.1a. This cartridge case displays both a circular impression left by the firing pin in the middle of the primer as well as breech face impressions left on the outer region of the primer not impressed into by the firing pin.

(a) Breech face of a shotgun barrel.

(b) Breech face impressions on cartridge case primer.

These breech face impressions are considered to be analogous to a firearm's "fingerprint" left on a cartridge case. As such, many believe that expended cartridge cases can be matched to the firearm from which they were fired by comparing an unknown source cartridge case to one of known source. Doing so would be useful for determining, for example, whether a questioned cartridge case was discharged from a suspect's firearm. The questioned cartridge case could be compared to another cartridge case that was known to have been discharged from the suspect's firearm. Such comparisons have long been performed by forensic examiners. However, the development of statistical methods for the field of forensic science has recently grown in interest (Council (2009)).

The Congruent Matching Cells (CMC) method was developed at the National Institute of Standards and Technology to perform cartridge case matching Song (2013). The method involves partitioning a cartridge case scan into a grid of "cells" and comparing these cells to associated regions in another cartridge case scan. To date, no open source implementation exists of the method exists. The goal of this project is to provide such an open source implementation, in the form of the `cmcR` package, so that the method, including its strengths and limitations, can be explored and scrutinized by a wider user base. The `cmcR` package puts a large emphasis on

providing the user flexibility in how they process the cartridge case data in every step of the CMC method. Decisions need to be made in how the cartridge cases are pre-processed before they are compared using the CMC method. There are also considerations to make when comparing the a pair of cartridge case scans such as the number of cells to partition a scan into. Finally, once comparisons are made, the user must define thresholds used to determine which cell pairs will be classified as "Congruent Matching Cells." The decisions made at each step of this method have a demonstrable impact on the final results. Further, little consensus is shared across the CMC literature as to what the "best" choices are. Thus, the `cmcR` package was developed to provide a wide swathe of options for exploration.

Chapter 3 provides background for the pre, inter, and post-processing options available in the `cmcR` package. Specifically, section 3.2 details the pre-processing procedures available. Section 3.3 explains two versions of the CMC method, the initially proposed and an "improved" version of the method, implemented in the `cmcR` package. Chapter 4 illustrates how to use the `cmcR` package in the form of a possible use case. Chapter 5 provides results and discussion of applying the `cmcR` package's functions to a set of 60 cartridge case scans commonly used as test cases in the CMC literature. This includes a discussion of the method's sensitivity to pre-processing decisions.

CHAPTER 2. REVIEW OF LITERATURE

The Congruent Matching Cells method as it would come to be called, was proposed by Song (2013). This initial proposition contained an overview of the key ideas motivating the method. In particular, the fact that a pair of matching cartridge cases may only have a handful of "valid correlation areas" shared between the two of them. These are defined to be regions of the cartridge case upon which the breech face impressed strongly and thus contain "individual characteristics of the ballistics signature." To make it easier to compare these valid correlation areas without being affected by invalid correlation areas, scans can be partitioned into a grid of "correlation cells." This effectively creates a more granular similarity metric; one that is more sensitive to smaller regions of extremely high similarity between two scans. A pair of scans will be judged a match or non-match based on a set of "identification parameters," which are features extracted from the comparison between two cells that should indicate similarity. To call a pair of cells "matching," these features would need to pass certain criteria. For example, the maximum cross-correlation function (CCF) value between two cells is one identification parameter. To call a pair matching, their maximum CCF value needs to be above some minimum threshold. See section 3.3 for what and how features are used. The results of an implementation of the CMC method applied to 2D optical images of cartridge cases were published by Tong et al. (2014). The results indicate that, after tuning the classification thresholds, they were able to differentiate between the known match and known non-match cartridge cases in their test set without error. Ott et al. (2017) provide further validation of the method by implementing it on two sets of cartridge cases from firearm examiner proficiency tests.

A few improvements of the method have been published since the initial proposal. Many of these improvements involve changing the way that extracted features are used to call a cell pair a match. Tong et al. (2015) propose an improvement that is implemented in the cmcR package. Briefly, the

proposed improvement involves extracting features from a pair of cells for a number of rotations of one of the cartridge case scans. However, rather than only considering the rotation at which a cell pair achieves maximum similarity as dictated by the initially proposed method, the features are compared across all rotations to determine whether the cartridge case scans are highly similar within a small range of rotations. See section 3.3.3 for more details. Chen et al. (2017) propose using the behavior of another feature extracted during the comparison process: the translations at which a cell pair achieves maximum CCF. In particular, this "convergence algorithm" they propose is based on the fact that the translations at which different cell pairs achieve maximum CCF will tend to agree once two matching cartridge cases are rotationally aligned. Non-matching cartridge cases, on the other hand, will likely not have such agreement across any rotation values. This convergence algorithm has not been implemented in the cmcR package.

Recently proposed improvements to the CMC method have attempted to solve an issue of dealing with missing data in a cartridge case scan. In particular, most cartridge case scans contain a considerable amount of "whitespace" that can be due to the imperfections in the scanning process. However, most of the missing data in a scan come merely from the fact that cartridge cases are normally circular while scan data are stored in rectangular arrays. Thus, a lot of missing data are seen in, for example, the corners of a scan. The issue is not helped, however, by the fact that the CMC method involves partitioning a scan into a grid of rectangular cells. It is often the case that a cell contains only a sliver of a cartridge case scan - especially if the partition of the scan is fine. However, such cells are given just as much of a "vote" as cells containing mostly non-missing values. Chen et al. (2018) propose a fix to this by considering the proportion of a cartridge case scan that has been classified as "congruent matching" to another scan, i.e., the "normalized congruent matching area," rather than just the number of congruent matching cells determined under the original method (or its improvements).

In the same vein as Chen et al. (2018), Tong et al. (2018) propose effectively re-weighting the cross-correlation function based on the number of missing values between a pair of cells. It is unclear whether this proposed "valid data based normalized cross-correlation" metric takes into

account whether a value is missing due to it being on the exterior of a cartridge case primer scan or due to imperfections in the scanning process.

CHAPTER 3. METHODS AND PROCEDURES

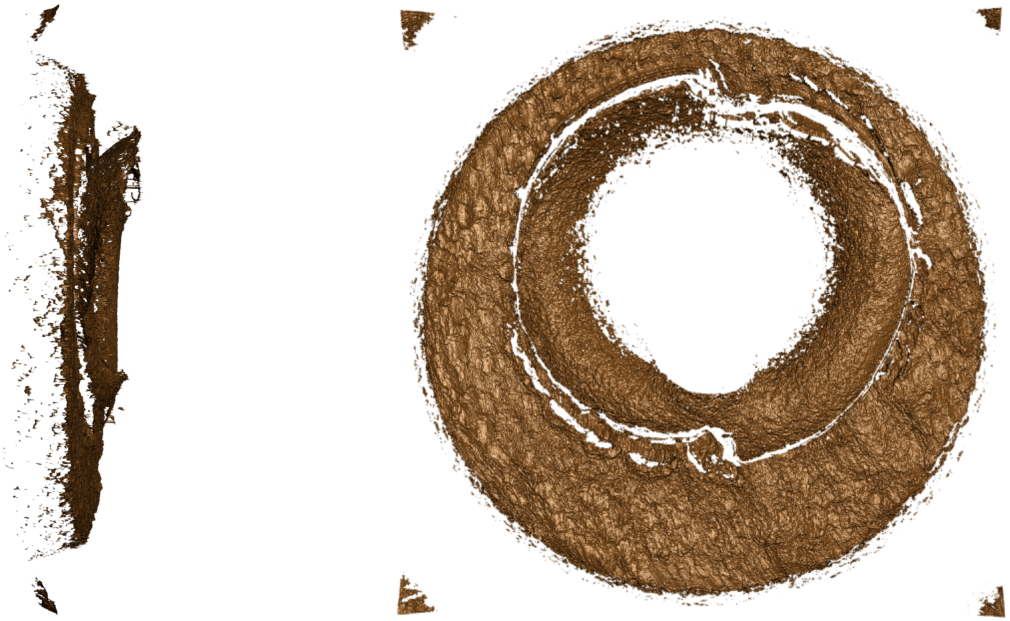
This section will describe the various methods used to implement the CMC method. This includes a discussion of the methods used to pre-process the cartridge case scans as well as a detailed description of the CMC method and one of its extensions.

3.1 Introduction

Cartridge case data commonly come in two forms: 2D grayscale images and 3D topographical scans taken via confocal microscopy. It has been shown to be easier to differentiate between matching and non-matching cartridge cases using 3D topographies (Tong et al. (2015), Tai (2019)). Cartridge case data from a variety of studies in either form are openly available for download on the NIST Ballistics Toolmark Research Database (Zheng et al. (2016)). This package was designed specifically for use with the 3D topographies.

In either form of the data, the cartridge case surface is represented as a *surface matrix* of pixels whose values correspond to the height of the cartridge case at a particular location. The 3D topographies are commonly stored in an .x3p (XML 3-D Surface Profile) file format that includes metainformation such as who took the scan and the parameters under which the scan was taken (e.g., the lateral resolution in microns) (see: https://sourceforge.net/p/open-gps/mwiki/X3p/). The `x3ptools` package in R provides an interface to manipulate and visualize these .x3p files (Hofmann et al. (2019)). Figures 3.1a and 3.1b show a left-hand and top-down view, respectively, of a cartridge case scan visualized using the `x3ptools::image_x3p` function. These figures demonstrate that the scans represent only the surface of the cartridge case; allowing for representation via a single matrix of values. Note that white regions in the images below represent unobserved or missing values. When read into R using the `x3ptools` package, these elements are encoded as NA. The size of a surface matrix depends largely on the lateral resolution with which the scans were taken.

For example, the scans used in Fadul et al. (2011), one of which is the scan shown in Figure 3.1, were taken with a lateral resolution of 6.25 microns per pixel. The actual surface matrices from this study vary around 1200x1200 pixels in size, the example below having dimensions 1231x1223.



(a) Left-hand side view

(b) Top-down view

Figure 3.1: Two views of a cartridge case scan

Only certain regions of a cartridge case come into contact with a firearm's breech face during the firing process. As such, cartridge scans must undergo some pre-processing before their breech face impressions can be compared. Figure 3.2 shows the same cartridge case from Figure 3.1 with regions that should be removed during pre-processing. The circle in the center highlights a plateaued region of the cartridge case, seen more prominently in Figure 3.1a, that is pushed aside by the firing pin and into the firing pin hole during the firing process. This region does not commonly come into contact with the breech face. The four clusters of observed values in the corners of the scan are artifacts of the staging area in which the scan was captured and thus are also not of interest. The task in pre-processing is to remove these unwanted regions from the scan to accentuate unique markings left by the breech face.

Figure 3.2: Cartridge case scan with highlighted regions to be removed during pre-processing.

3.2 Surface Matrix Pre-processing Procedures

Before using the CMC algorithm to determine whether two cartridge cases are a "match," each image must undergo some standardization pre-processing. The purpose of these steps is to ensure that no superfluous information is used to classify a pair as a match or non-match. Many authors do not discuss their pre-processing procedures in great detail and there is considerable evidence that the final results are highly sensitive to how the scans are pre-processed see section 5.

In their raw format, breech face scans contain additional information not useful in improving the ability of the CMC algorithm to differentiate between known matches and known non-matches. Figure 3.3 shows an unprocessed cartridge case scan. Data from such scans are commonly stored in a "top-down" perspective. That is, the row/column indices of a surface matrix play the role of "length" and "width" while the actual elements of surface matrix represent height values. In the images below, the height is represented in grayscale where whiter pixels are associated with larger height values. Purely white pixels (255 intensity) represent NA or missing data.

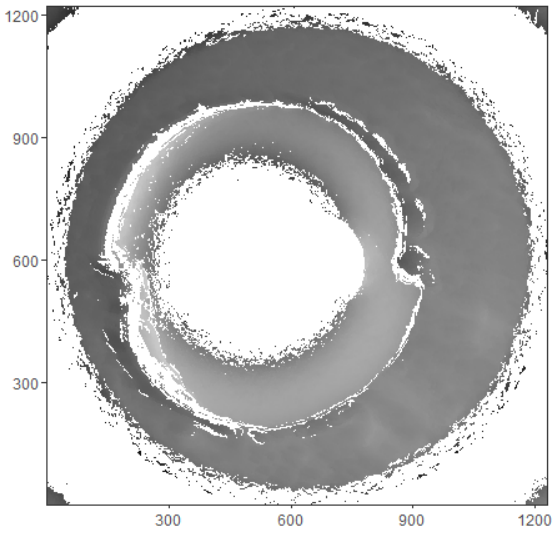


Figure 3.3: An unprocessed cartridge case scan

3.2.1 Breech Face Selection

One of the pre-processing goals is to segment-out the actual breech face impressed region of the cartridge case scan. A variety of methods (e.g., Watershed, RANSAC, Circular Hough Transform) were tested to determine the most effective way of extracting the breech face impression. RANSAC was ultimately determined to be the most consistently effective and computationally fast method among those tried.

3.2.1.1 RANSAC Method

For our purposes, the RANSAC (RANdom SAmple Consensus) method is an iterative method for fitting a 2-dimensional plane to a set of data in the presence of outliers or, in our case, multimodal data (Fischler and Bolles (1981)). An illustration of the method for 2-dimensional data is shown in Figure 3.4 (Gallostra (2017)). In the illustration, the goal is to fit a line through the data points shown in the top left box that has a high number of data within close proximity of it. This is clearly a line that begins in the bottom left corner of the data and moves to the top right corner. To fit the data, two points are randomly sampled from the data (top left box) and a candidate line connecting the two points (top right box) is "scored" (middle left box) by counting the number of

data lying within some distance of the line (the best distance is subject to experimentation). The number of data lying within this threshold of the line are referred to as *inliers*. This process is repeated numerous times (middle right and bottom left) and the line with the highest score (highest number of inliers) is selected (bottom right).

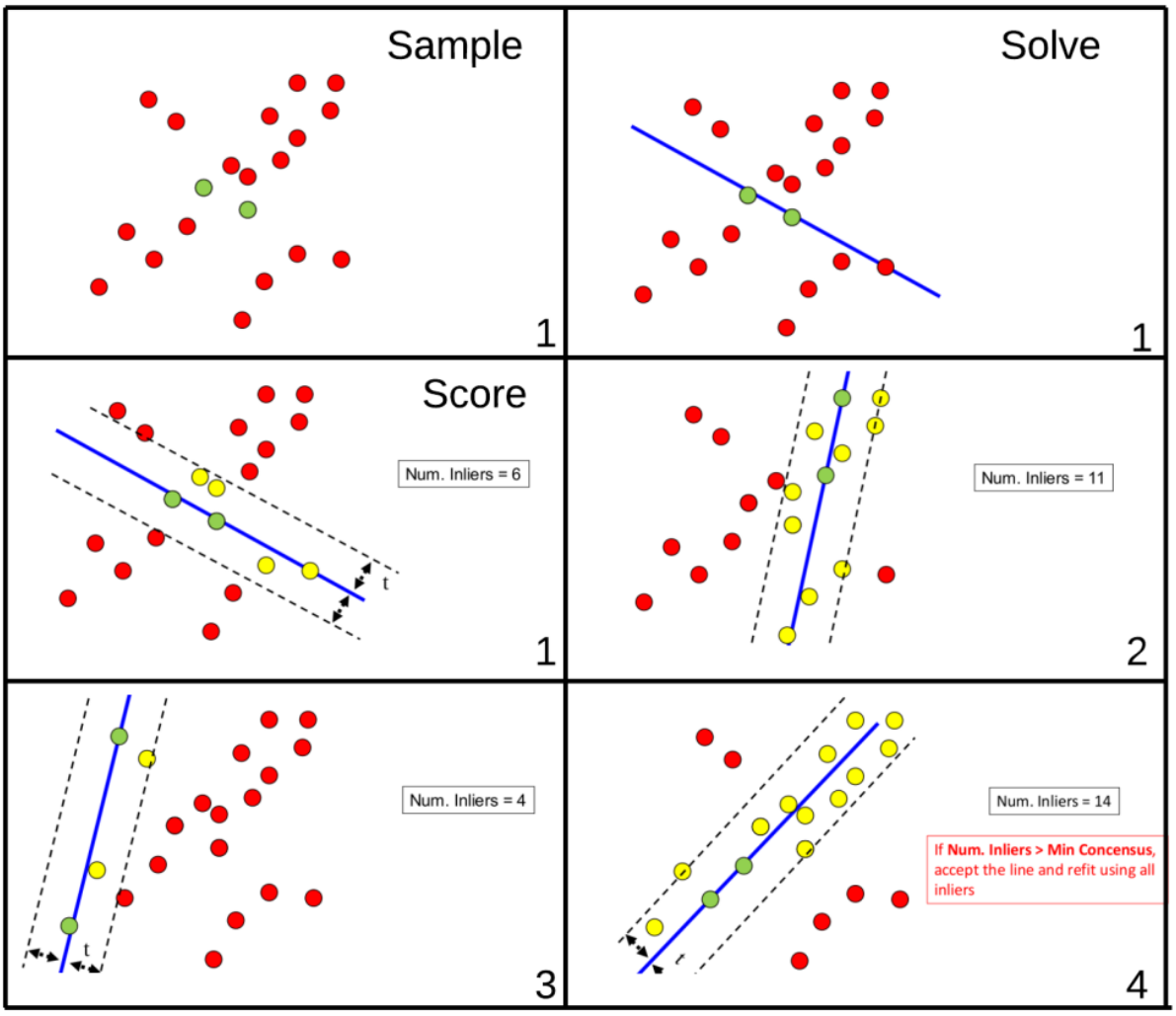


Figure 3.4: Illustration of the RANSAC method.

The RANSAC method applied to 3-dimensional data is analogous. Three data are randomly selected through which the candidate plane is fit and scored at each step of the algorithm. We can consider observed values that are within, say, 2 microns of this fitted plane to be part of the breech

face impressed region of the cartridge case. In particular, we can consider the residuals between the fitted plane and these observed values. Figure 3.5 shows the resulting residuals between the scan's observed values and the fitted plane.

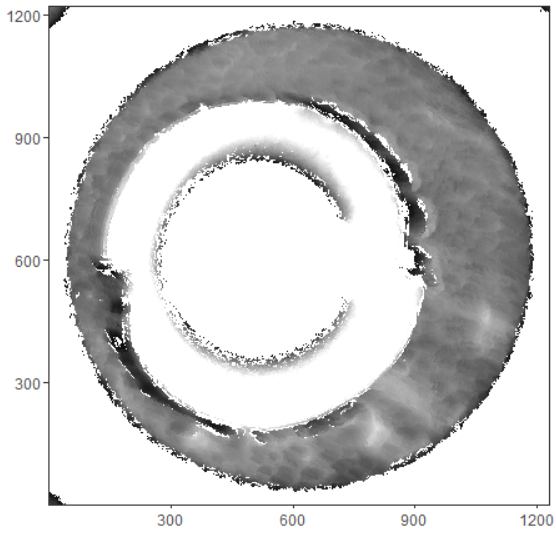


Figure 3.5: Surface matrix returned by the RANSAC method.

Although we have now removed a considerable amount of unnecessary information, there are still remnants of the firing pin impression ring in the center of the scan. We can also see a small cluster of pixels left in the top left corner of the scan. These are removed during further pre-processing described below.

3.2.1.2 Surface Matrix Cropping

The easiest of the two problems to solve is removing the small cluster of elements left in the corner of the scan. We can simply crop out the unnecessary rows/columns outside of the breech face. This will have the added bonus that each cell in the CMC algorithm will contain a larger proportion of observed breech face impression pixels, especially on the edge of the breech face impression. Without cropping the unnecessary exterior rows/columns, many of the edge cells would contain mostly if not all NA values, thus making them useless in determining whether two cartridge cases match.

The method we use to automatically identify the rows/columns to crop out is fairly simple. There are likely more robust ways of solving this problem, but this seems to work well for the 40 cartridge case scans used in Tong et al (2015). For a given surface matrix, we want to find to identify the rows and columns containing the first/last pixel of breech face impression data. Finding these rows/columns amounts to essentially drawing a tight box around the breech face impression. To accomplish this task, we look at the rows and columns in the middle of the image as shown in the image below. To determine which columns to crop out, we identify the left and right-most columns containing a non-NA pixel value based on the middle rows as shown in the image below. A similar process is performed to determine which rows to crop out. Any rows/columns whose indices fall between the identified extreme rows/columns are kept. An illustration of the rows/columns considered when cropping the surface matrices is shown in Figure 3.6.

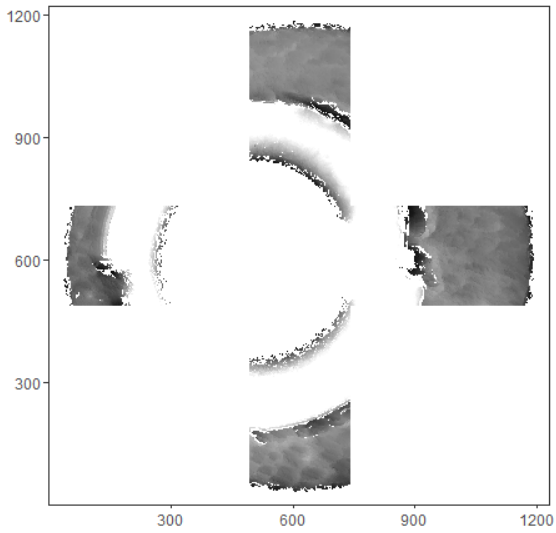


Figure 3.6: Region of the surface matrix considered when cropping.

3.2.1.3 Firing Pin Radius Detection

The second filtering problem to tackle is in removing the region of the firing pin impression ring that happens to have the same height values in the original cartridge case scan as the RANSAC-selected breech face impression region. Similar to the image cropping method described above, there

are likely a number of methods that could be used to identify and remove this "minutiae." A number of methods were tried (e.g., Watershed, Circular Hough Transform) and a Hough Transform-based method was determined to be the most successful.

Firing pin circumferences are not homogenous across all types of firearms. Further, small changes in the way that a cartridge case is loaded into a barrel cause the firing pin to strike the primer in different locations, even across consecutively fired shots. This motivates the need for an automatic method to detect the location of a firing pin ring in a cartridge case scan.

Circular Hough Transform The Hough Transform is a consensus-based shape detection algorithm. The Hough Transform can be used to detect a variety of shapes in images, yet we are interested in using it to detect circles. In terms of Cartesian coordinates, a circle is characterized by its center and its radius. Given a radius, the Hough Transform algorithm will try to detect the center of a circle with the given radius in an image (Hough (1962), Stockman and Shapiro (2001)).

The algorithm is best explained through an example like the one given in Figure 3.7 (Commons (2014)). Suppose we wanted to detect the red circle in the image on the left. Suppose further that we knew the radius r of the circle but desired to use the Hough Transform to find its center (x, y). The motivation for the algorithm goes as follows: if we correctly sample a point that lies on the circle we want to detect, then the center should be exactly r units away from the sampled point. However, considering only one sampled point doesn't help determine *where* the center is relative to that point. Thus, at a particular sampled point, the algorithm considers all points that are r pixels away as "potential" centers. We can think of this locus of pixels as "votes" cast by the sampled point for what might be the true center of the circle. The 4 white points around the red circle represent 4 such sampled points. The algorithm stores the votes in an *accumulator array* for all sampled points. The blue image on the right of Figure 3.7 is a visual representation of the accumulator array for the 4 white sampled points. The 4 light blue circles represent the locus of points that are r pixels away from the 4 white points, respectively. Although there is little intersection between the 4 circles, we can see that there is clearly one red pixel that all 4 sampled

points voted for as a potential center. This consensually voted pixel is then the natural choice for the true center of the circle.

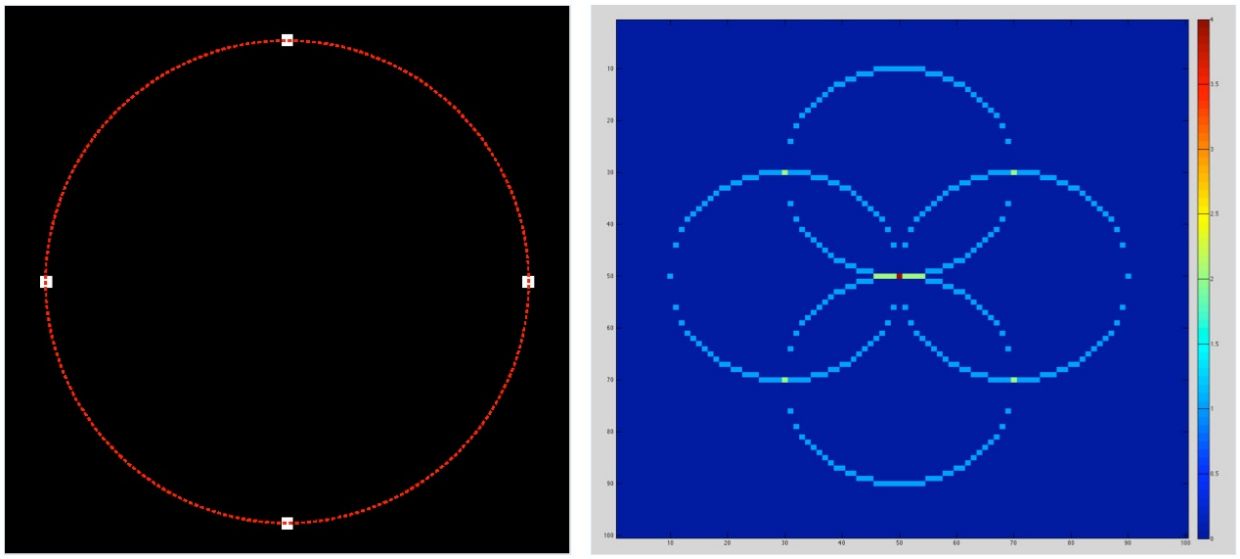


Figure 3.7: Illustration of the circular Hough transform.

Figure 3.7 is simplified in the sense that the only non-zero black pixels in the image were part of the circle. Thus, sampling from the set of non-zero pixels would assuredly mean sampling from the circle we want to detect. This is not so simple in most realistic examples. Considering the breech face impression scans, sampling from the set of non-NA pixels would likely yield pixels in the interior of the impression. Instead, we want to sample points along the firing pin impression ring. Using the process of *edge detection*, we can make the firing pin impression ring more distinct from the rest of the breech face impression, thus making it easier to identify to the Hough Transform. A Canny edge detector followed by a Hough Transform is used to identify the firing pin impression ring in this package (Canny (1986)).

Firing Pin Impression Circle Radius Detection Another way in which the red circle example is overly simplified is in the assumption that we know that radius of the circle to be detected ahead of time. Firing pin circumferences vary across firearms, so it is naïve to assume that we will know the firing pin radius, especially for new cartridge cases of unknown source. This

section will detail a method of automatically detecting the firing pin impression radius within an image like the one shown below that has been successful barring a few exceptions.

To identify the diameter of the inner circle in Figure 3.5, we can count the number of the number of non-NA pixels in each row/column of the matrix and identify where this count attains local maxima. An example of the non-NA row counts is shown in Figure 3.8 for surface matrix shown in Figure 3.5. We can see that the count is rather jagged across the rows, most likely due to the fact that there are small regions of NA pixels within the breech face impression scan. However, there are clearly two modes that we would like to identify. The second plot shows the result of passing a moving average smoother over the count values to make the local maxima easier to identify. We can find the maxima simply by first-order differencing the count (similar to how one would with a time series) and determining where these differences change from positive to negative (i.e., where the derivative changes from positive to negative).

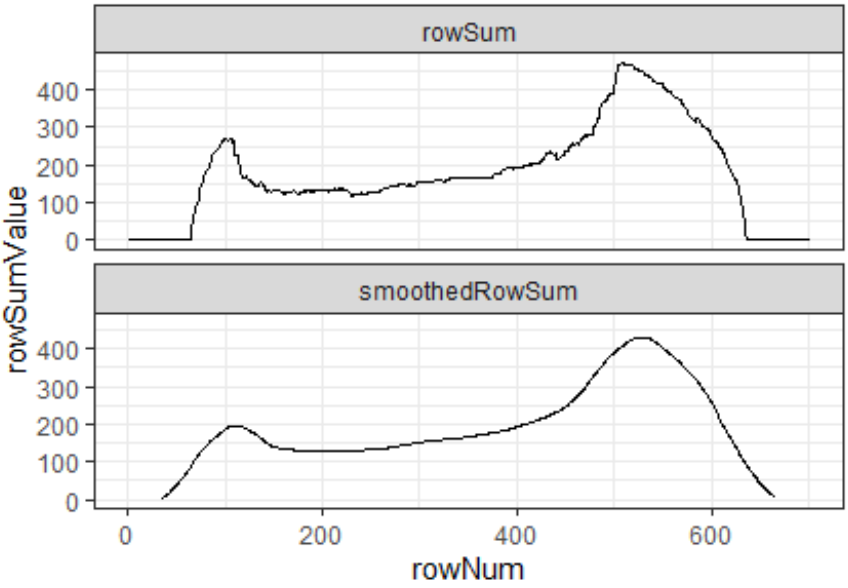


Figure 3.8: Non-NA row sums for the surface matrix shown in the figure above.

The distance between the rows at which the non-NA pixel count achieves its two largest local maxima is an estimate of the diameter of the firing pin impression circle. We can obtain a number of estimates by repeating this process for various rotations of the cartridge case scan. In the current

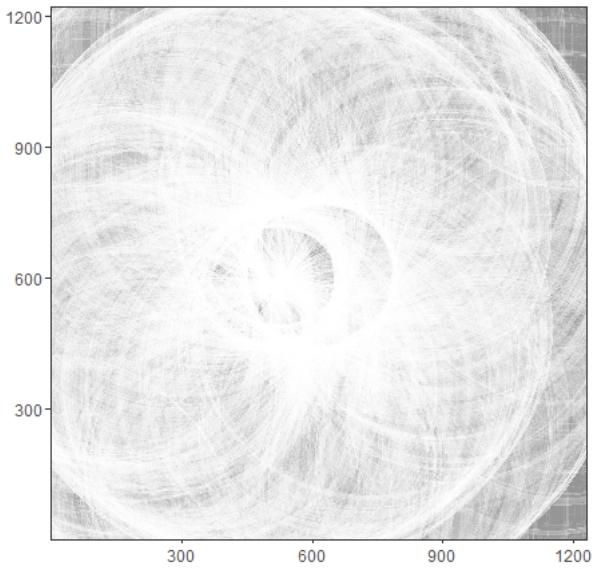


Figure 3.9: A circular Hough transform accumulator for radius value 402.

implementation, this local maxima method is repeated for rotations by $\theta \in \{0, 15, 30, 45, 60, 75\}$ degrees. The estimates are then summarized in some way, currently by taking their mean, to come up with a final diameter estimate. By giving half of this diameter estimate to a Hough Transform, we can determine an approximate center of the firing pin impression circle. Figure 3.9 shows a circular Hough transform accumulator for a cartridge case scan with associated radius $r = 402$. A variety of radius values are tested around the estimate obtained from the local maxima method. A final circle center is selected based on a heuristic described below.

As discussed, every possible circle center considered by a Hough transform is given a "score" that indicates the number of sampled points that voted for it. This can either be raw count or a standardized score, say, between 0 and 1, measuring the level of confidence that a particular center point is the true center. Such standardized scores are returned by the `imager::hough_circle` function (Barthelme (2019)). For each radius value, it's natural to select the center with the highest associated score as the estimated center. The plot below shows the maximum Hough scores for various radius values. The heuristic used to settle upon a final center point is based on this series of Hough scores. In particular, we search in this series for the longest consecutive run of radius values with consistently high associated scores (defined to be anything above some high

quantile of Hough scores). In the plot below, we see a run of radius values from 201 to 206 that are consistently high. We then use the (floored) median of these values as the final radius estimate and the center associated with this radius value as the final center estimate.

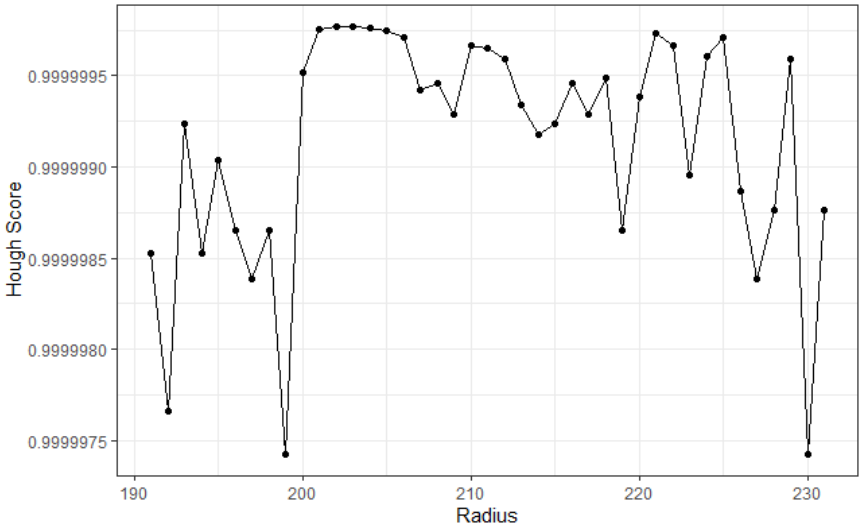


Figure 3.10: Plot of the maximum Hough scores for various radius values.

Figure 3.11 shows the breech face impression after cropping out exterior whitespace and filtering pixels inside of the Hough transform estimated circle.

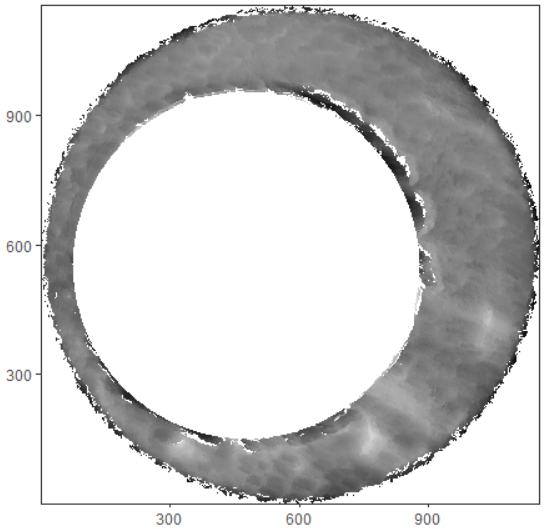


Figure 3.11: Selected surface matrix after cropping and removing the firing pin impression.

3.2.2 Surface Matrix Resizing

It is common in the CMC literature to resize a cartridge case scan surface matrix to a smaller dimension. This is meant to improve computational efficiency. A logical way to downsample these surface matrices is simply to start with, say, the element in the bottom-left corner of the matrix and travel along each row/column of the matrix extracting every other element. This type of downsampling is performed in the `x3ptools:sample_x3p` function (Hofmann et al. (2019)). This is illustrated in the example below.

$$\begin{bmatrix} \vdots & \vdots & \vdots & \iddots \\ 7 & 8 & 9 & \dots \\ 4 & 5 & 6 & \dots \\ 1 & 2 & 3 & \dots \end{bmatrix} \rightarrow \begin{bmatrix} \vdots & \vdots & \iddots \\ 7 & 9 & \dots \\ 1 & 3 & \dots \end{bmatrix}$$

We can see that the output matrix would be a fourth the size of the original matrix. Such a downsampling scheme is used by Chen et al. (2017) to reduce the size of the cartridge case scans' surface matrices. The scans used in this paper, from Fadul et al. (2011), are around 1200×1200 prior to downsampling and thus around 600×600 after. It is unclear how other authors downsample their images. In image processing, images can be sampled to an arbitrary dimension using a variety of techniques all related to the idea of *image convolution* and interpolation. For our purposes, a possible downside of many of these techniques is that they require taking local averages of the pixel intensities in the image. Thus, downsampling to an arbitrary dimension means that we will be tampering with the original, observed surface matrix values. Nonetheless, the `cmcR` package allows the user resize the cartridge case surface matrices using a variety of resizing techniques, as discussed in the the following section.

Resizing via Convolution Example Intuitively, we'll think of image convolution as combining two different images, one typically much larger than the other, in some way to change the pixel intensities in the large image. The example in which we downsample a matrix by taking the

element in every other row/column, starting in the bottom left corner can be interpreted within the more general framework of image convolution. In fact, we can interpret many of the processes implemented by the `cmcR` package, including Gaussian filtering discussed in section 3.2.3.1 and calculation of the cross-correlation function discussed in 3.3.1, as essentially performing convolution between two different matrices. Thus, it is worthwhile to build up some intuition for how this process works. See section 3.2.3.1 for an illustration of convolution can be implemented in the frequency domain to perform Gaussian filtering. For this section, we will focus on convolution in the spatial domain.

Returning now to the example, consider the following matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. We'll refer to matrix $\boldsymbol{B}$ as a *kernel*.

$$\boldsymbol{A} = \begin{bmatrix} 5 & 1 & 2 \\ 2 & 8 & 1 \\ 9 & 7 & 4 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 3 & 6 \\ 1 & 5 \end{bmatrix}$$

The process of *convolving* matrix $\boldsymbol{A}$ with matrix $\boldsymbol{B}$ can be thought of as overlaying $\boldsymbol{B}$ on groups of elements within $\boldsymbol{A}$. A linear combination of the $\boldsymbol{A}$ elements are then calculated with coefficients equivalent to the associated elements in $\boldsymbol{B}$. For simplicity, we'll start in the top left corner of matrix $\boldsymbol{A}$ and move from left to right. The calculation of the $[1, 1]$th element of the new convolved matrix, $\boldsymbol{A} * \boldsymbol{B}$, would look something like the following.

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} 5 \cdot 3 + 1 \cdot 6 + 2 \cdot 1 + 8 \cdot 5 & \_ & \_ \\ \_ & \_ & \_ \\ \_ & \_ & \_ \end{bmatrix} = \begin{bmatrix} 63 & \_ & \_ \\ \_ & \_ & \_ \\ \_ & \_ & \_ \end{bmatrix}$$

Then for the $[1, 2]$th element, the following would be calculated.

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} 63 & 1 \cdot 3 + 2 \cdot 6 + 8 \cdot 1 + 1 \cdot 5 & \_ \\ \_ & \_ & \_ \\ \_ & \_ & \_ \end{bmatrix} = \begin{bmatrix} 63 & 28 & \_ \\ \_ & \_ & \_ \\ \_ & \_ & \_ \end{bmatrix}$$

and so on. Performing convolution in this manner makes the resulting matrix take on properties of both matrices. If, for example, the kernel matrix's elements sum to 1, then we can think of each step of the convolution process as taking a weighted average of the neighborhoods in the larger matrix. In this case, if an element in a matrix is significantly larger than the elements surrounding it, then locally averaging will dampen or "smooth" this element around its neighbors. For any two images A, B, the convolution is defined to be

$$(A * B)[m, n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} A[i, j]B[m - i, n - j]$$

where, in practice, the sum would be assumed 0 outside of the greater of the two images' dimensions.

Not that the calculation of the $[3, 1]$th element, or any elements in the bottom row/column of $\boldsymbol{A} * \boldsymbol{B}$, would not be possible with the current way we are calculating convolution since certain elements of $\boldsymbol{B}$ would not have associated $\boldsymbol{A}$ elements. The "boundary" cases are often dealt with by specifying boundary conditions such as

1. Pad the matrix with 0s (Dirichlet)
2. Pad the matrix with additional copies of the boundary rows/columns (Neumann)
3. "Wrap" the matrix's boundary back to the other side. Equivalent to mapping the matrix onto a torus (Periodic).

The type of boundary conditions used is highly problem-specific. When dealing with the cartridge case scans, the `cmcR` package first subtracts from each element of the surface matrix the matrix's average value. Then, Dirichlet boundary conditions are used in which the matrix is padded with 0s.

So far this example has not actually discussed how convolution can be used to resize an image. A plethora of resizing methods exist. A simple implementation using the principles of convolution

is the following (Wang (2016)). Consider the following matrix.

$$\boldsymbol{C} = \begin{bmatrix} 2 & 10 & 8 & 1 \\ 5 & 5 & 98 & 0 \\ 3 & 1 & 4 & 9 \\ 0 & 4 & 9 & 2 \end{bmatrix}$$

Suppose we were interested in downsizing this matrix to be 2×2. One possible way to do so would be to simply extract every other element along the rows/columns of the matrix. There are 4 different 2×2 matrices that we could extract depending on where we began this process. However, we may want the 2×2 to, in some way, maintain some of the characteristics of the original matrix. For example, the $[2, 3]$th element is significantly different from the other elements in the matrix, so we may want this element's "information" to still be represented in the downsampled matrix. One way we can do this is by convolving $\boldsymbol{C}$ with a simple 2×2 local averaging matrix of the form

$$\boldsymbol{D} = \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

The convolution, assuming again we start in the top left corner of $\boldsymbol{C}$ and consider, say, 0-padded boundaries, would be

$$\boldsymbol{C} * \boldsymbol{D} = \begin{bmatrix} 5.5 & 20.25 & 26.25 & .25 \\ 3.5 & 27 & 27.25 & 2.25 \\ 2 & 4.5 & 6 & 2.75 \\ 1 & 3.25 & 2.75 & .5 \end{bmatrix}.$$

We could then select every other row/column from this matrix to populate our downsampled matrix. We can see that in performing this local averaging, the extreme value of the $[2, 3]$th element of $\boldsymbol{C}$ has been spread to neighboring elements. This is clearly a simplified example. In real implementation, this moving average interpretation of convolution is computationally infeasible. Instead, a frequency-based approach is more often used. See section 3.2.3.1 for an example of how convolution can be used to perform Gaussian filtering.

3.2.3 Surface Matrix Filtering

Gaussian filtering is commonly performed in image processing and is often applied in the CMC literature to reduce the effects of unwanted characteristics of a cartridge case scan. Philosophically, however, one could argue that working with these surface matrices should not be considered as processing images and that it is dangerous to change the "raw" scan data too much during pre-processing. Nonetheless, the `cmcR` package allows the user to apply a Gaussian filter to the selected breech face impressions, if desired.

3.2.3.1 Gaussian Filtering

Gaussian filters can be used be used to reduce the effects of unwanted "signal" in a surface matrix. For our purposes, there are two types of signal that may detract from our ability to match two cartridge case scans based on breech face impressions. One type of signal that may exist in a cartridge case scan is large-scale form specific to a particular cartridge case. For example, imperfections in the manufacturing process could have caused one cartridge case to have flatter primer surface than another cartridge case. These large-scale, "low frequency" differences, having nothing to do with the breech face impressions left on the primer, could still deflate the overall cross-correlation function used to compare the two cartridge cases. On the other hand, imperfections in, for example, the scanning process yield a certain amount of "noisiness" in the surface matrix of the scan that doesn't exist on the actual surface of the cartridge case primer. We can classify this noise as small-scale, high frequency signal that could also throw-off our cross-correlation function values. A Gaussian filter is a tool used in image processing, among other places, to reduce the effects of unwanted signal on either or both scales. A *low pass filter*, analogous to passing a moving average kernel over an image, works to reduce the effects of high-frequency signal. The output of such a filter typically looks like a blurrier version of the input image. In contrast, a *high pass filter* aims to reduce the effects of low-frequency signal, thus making an image often appear "sharper." One way to implement a high pass filter (the one used in the `cmcR` package) is to first apply a low pass filter to an image and then subtract the resulting filtered image from the original. This is analogous to

"de-trending" a time series by subtracting away an estimated mean function. A *bandpass filter* is simply a filter that implements both a low and high pass filter. In the `cmcR` package, a bandpass filter is implemented by first applying a high-pass filter followed by a low pass filter; although other types of implementations exist (Greene (2020)). The example below is meant to help build intuition for how a Gaussian filter affects an image as well how the spatial and frequency representations of images are related (Bourke (1998)).

Gaussian Filter Example To gain some intuition into how Gaussian filtering affects an image, we will consider a simplified example. Consider the Figure 3.12 created by taking an entirely black (0 pixel value) 100×100 matrix and adding to it a white (255 pixel value) 20×20 box in its center along with $N(0, 20)$ white noise.

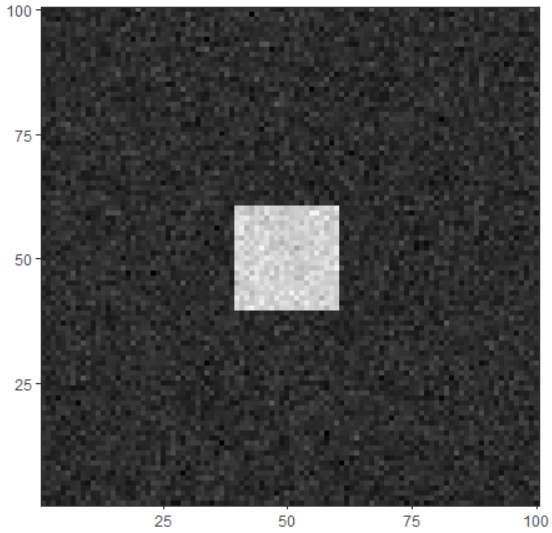
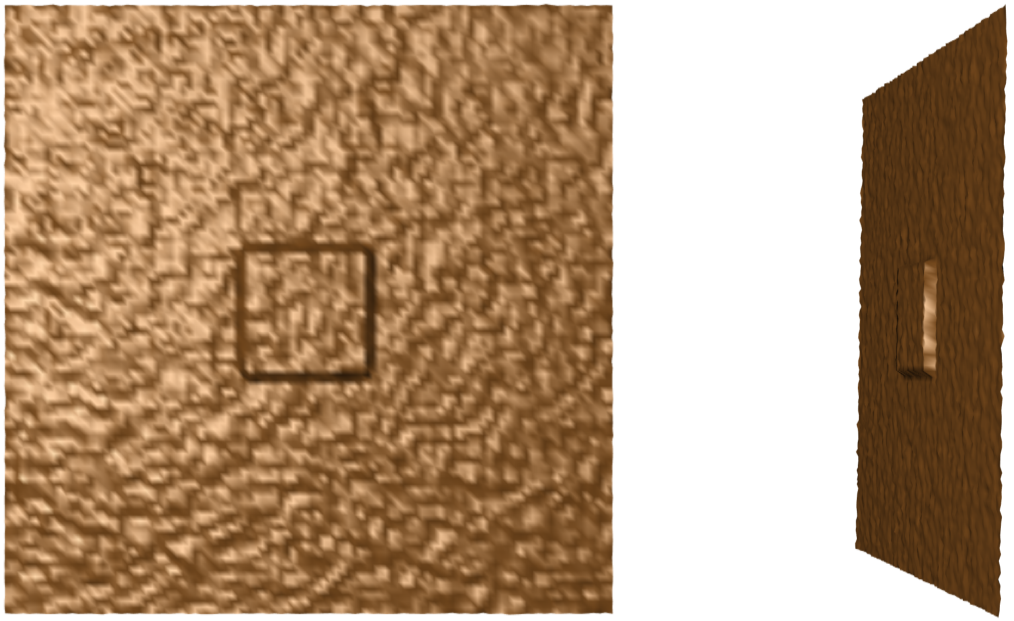


Figure 3.12: 100×100 matrix with a white box and Gaussian noise added.

For the sake of understanding the Gaussian filter, it will easier if we consider this image in 3 dimensions, where the image's pixel intensity value will be mapped to the height. Figure 3.13 shows a top-down and right-hand side view of the image shown in Figure 3.13. We can see that the white box in the middle becomes a large-scale plateaued region in the center while the Gaussian noise becomes small-scale waviness.



(a) Top-down view (b) Right-hand side view

Figure 3.13: Two 3D views of the image shown in Figure 3.12

While intuitive to think about a Gaussian filter as passing a moving average kernel over an image, such an implementation is computationally intensive in-practice. Instead, filtering is commonly done using the Convolution Theorem: for two bivariate, real-valued functions f, g, $f * g = \mathcal{F}^{-1}\left(\mathcal{F}(f) \cdot \mathcal{F}(g)\right)$ where $\mathcal{F}(\cdot)$ denotes the Fourier transform operator. The Fourier Transform of a bivariate, real-valued function $f : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$\mathcal{F}(f)(x, y) := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(u, v) e^{-i2\pi(xu+yv)} \, dudv.$$

We can think of the Fourier Transform as projecting the function f onto an (uncountably) infinite-dimensional orthonormal basis of sinusoids. Analogous to, say, simple linear regression in which an orthogonal polynomial basis is used to explain variation in a dataset, this basis of sinusoids can be used to perfectly explain all variation in the function f. For our purposes, it is impossible/unnecessary to apply the true Fourier Transform to a cartridge case surface matrix. Surface matrices are discretized representations of the true cartridge case surface, so we need to consider the discrete, finitely supported analogue of the Fourier Transform, the Discrete Fourier Transform

(DFT). For an $N \times N$ matrix A, the DFT of A is given by

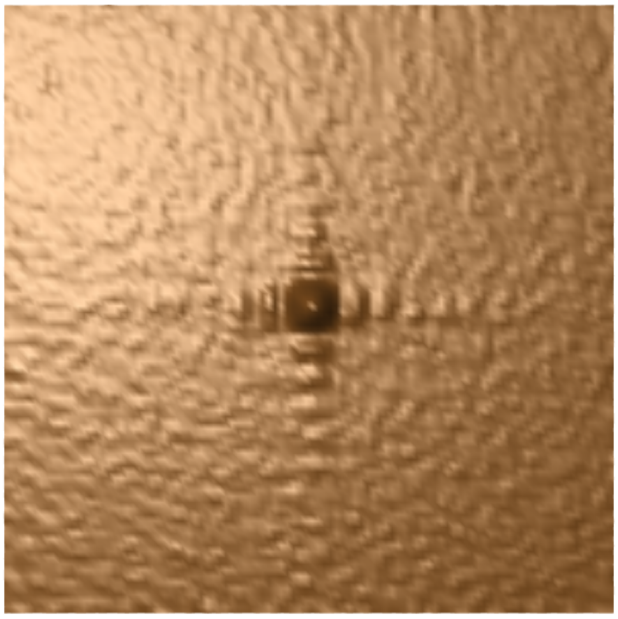
$$\mathcal{F}(A)[m,n] = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} A[i,j]e^{-\frac{2\pi i}{N}(jm+kn)}.$$

The Fast Fourier Transform is a particular implementation of the DFT that is commonly used since computing the DFT from the definition can be computationally cumbersome.

The Convolution Theorem says that convolution can be accomplished by performing pointwise multiplication in the frequency domain and then applying an inverse Fourier transformation. For intuition on this, Figure 3.14 shows the top-down and right-hand side views of the frequency domain representation of the image shown in Figure 3.12. The height values in these images now represent the (modulus of) coefficients of a discretized continuum complex-valued, sinusoidal basis functions. The center of the image represents the origin. Points near the origin correspond to sinusoids with low frequency (i.e., large wavelength). Points further from the origin then correspond to sinusoids of higher frequency. The larger height values near the origin indicate that low-frequency sinusoids are needed to explain the variability in the original image. In particular, these low-frequency sinusoids are needed to represent the white box in Figure 3.12. In contrast, the high frequency noise can only be explained by high frequency sinusoids. However, no one high frequency sinusoid is particularly effective at explaining all of the noise, so many of them are needed. This fact is represented in Figure 3.14 by the many small height values far from the origin.

Gaussian filtering in the frequency domains involves pointwise multiplying the frequency domain representation of an image with the frequency domain representation of a bivariate Gaussian density of a particular standard deviation. The size of this standard deviation affects how "blurry" the resulting image appears. As one might expect, a larger standard deviation leads to a more heavily blurred image. Figure 3.15 shows the frequency domain representation of a bivariate Gaussian density, specifically with a covariance matrix of $10\boldsymbol{I}_{2\times 2}$. While difficult to see, the height values far from the origin approach but never actually reach 0. This means that applying a Gaussian filter never truly removes signals, but only reduces how much they are represented in the original image.

We can now start to imagine what would happen if we performed pointwise multiplication between the images shown in 3.14a and 3.15a. Namely, height values closest to the origin, i.e.,



(a) Top-down view

(b) Right-hand side view

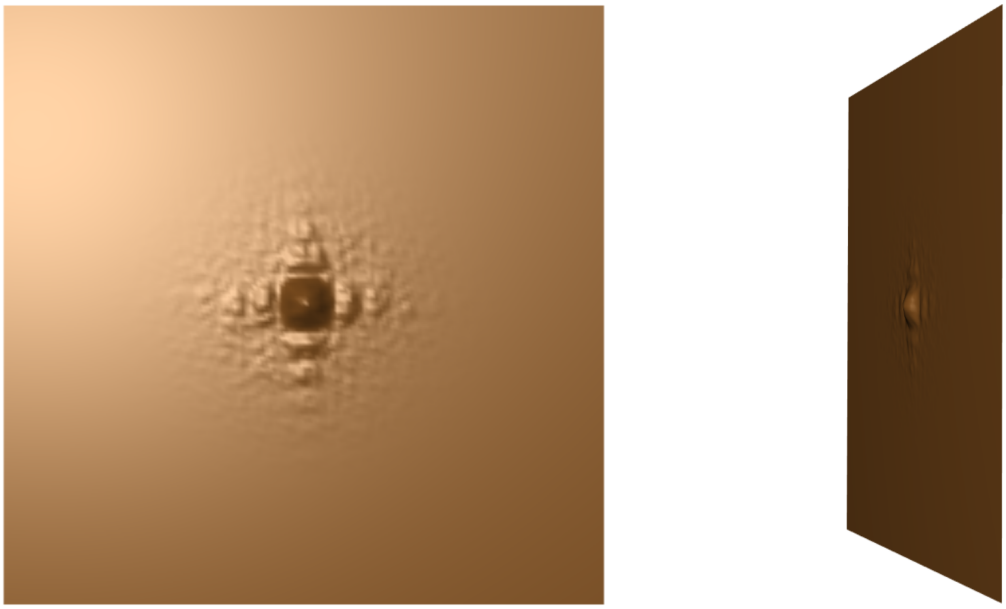
Figure 3.14: Two views of the frequency domain representation of the image shown in Figure 3.12

(a) Top-down view

(b) Right-hand side view

Figure 3.15: Two views of the frequency domain representation of a bivariate Gaussian density.

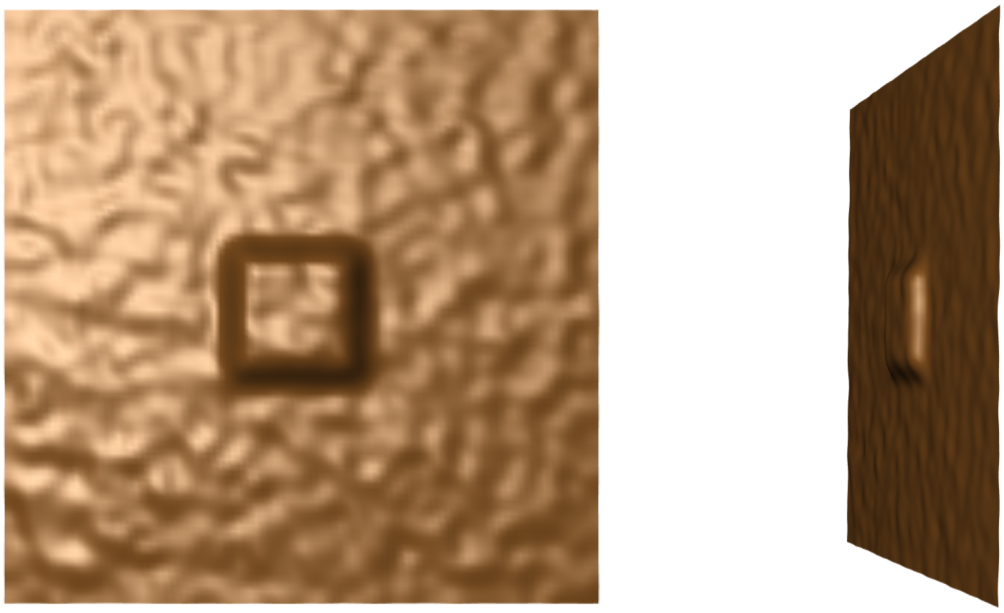
the low frequency signals, would be mostly preserved while those far from the origin would be heavily attenuated. Thus, this pointwise multiplication is equivalent to performing a low-pass Gaussian filter on the original image. Figures 3.16 and 3.17 shows the frequency and spatial domain representations of the filtered image. We can see that much of the noise in the original image has been reduced while the box has retained a good deal of its structure.



(a) Top-down view (b) Right-hand side view

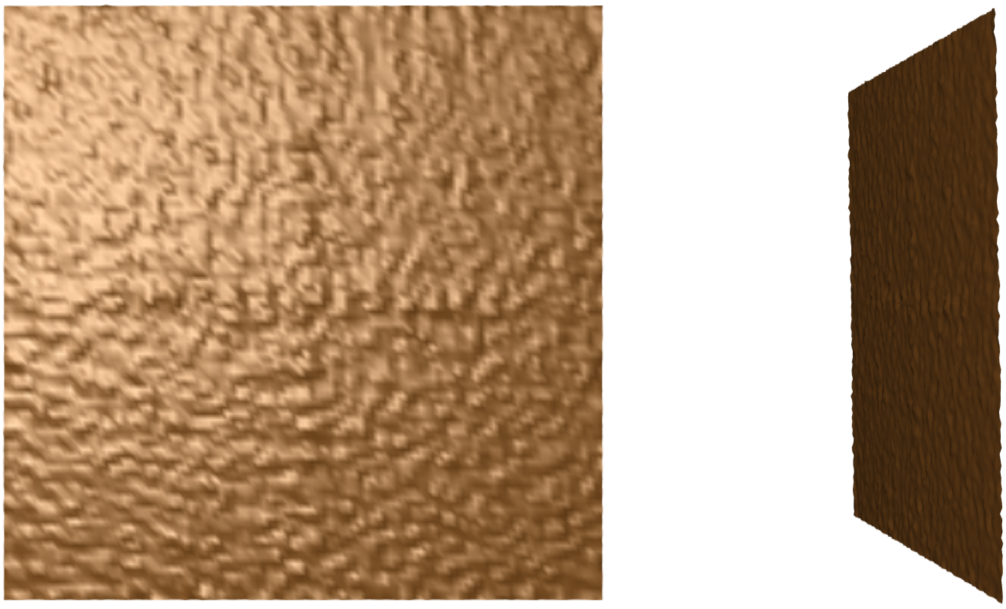
Figure 3.16: Two views of the frequency domain representation of low-pass filtered image

Subtracting the image in Figure 3.17 from the original image is equivalent to performing a high pass Gaussian filter. The frequency and spatial representations of the high pass filtered image are given in Figures 3.18 and 3.19. We can see in Figure 3.18 that removing the complementary low pass filtered image from the original has severely reduced the coefficients associated with low frequency sinusoids. In doing, so we can see in Figure 3.19 that the box is effectively unnoticeable in the filtered image relative to the noise surrounding it. Some semblance of the box's edges can still be seen in the filtered image, but we might expect this since these edges would need to be explained with higher frequency sinusoids than the flat part of the box.



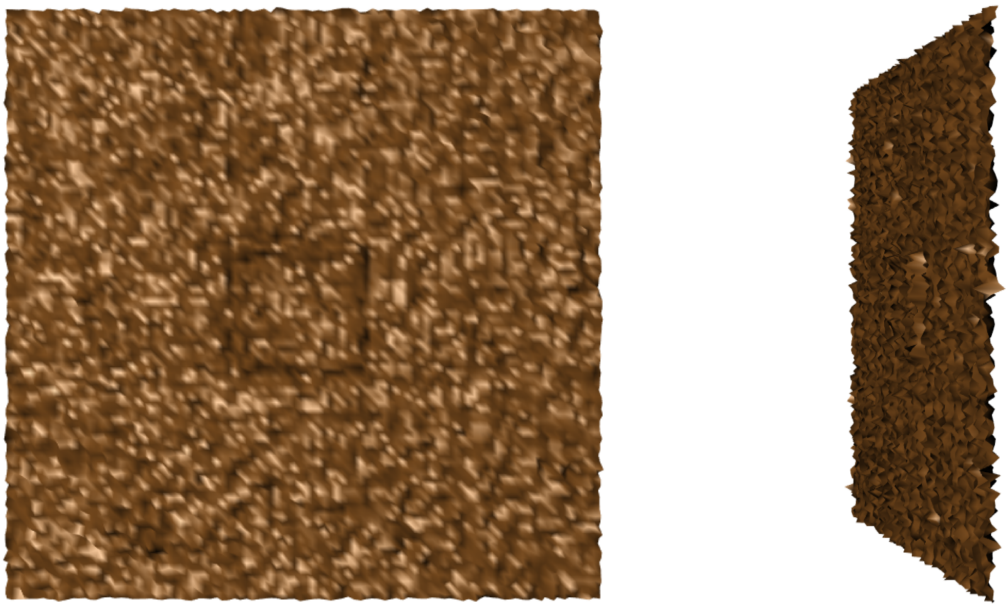
(a) Top-down view (b) Right-hand side view

Figure 3.17: Two views of the spatial domain representations of low-pass filtered image.



(a) Top-down view (b) Right-hand side view

Figure 3.18: Two views of the frequency domain representation of high-pass filtered image



(a) Top-down view (b) Right-hand side view

Figure 3.19: Two views of the spatial domain representations of high-pass filtered image.

3.3 Congruent Matching Cells method

The Congruent Matching Cells (CMC) method was developed at the National Institute of Standards and Technology (NIST) to quantify the similarity between two spent cartridge cases based on their breech face impressions (Song (2013)). The CMC method involves dividing a breech face impression scan into a grid of cells and comparing each cell in one scan to a corresponding region in the other scan. Figure 3.20 illustrates this idea. The motivation for this particular method is that two breech face impressions tend to have regions of high similarity, e.g., where the firearm's breech face impressed strongly into the cartridge case, and regions of low similarity, e.g., where the breech face may not have come into contact with the cartridge case. The highly similar regions may be drowned-out, in a sense, by the less similar regions if one were to calculate a similarity score considering the entirety of the two scans. By breaking up the breech face scans into a grid of cells, one can instead use the number of highly similar cells between the two scans as a more granular similarity metric.

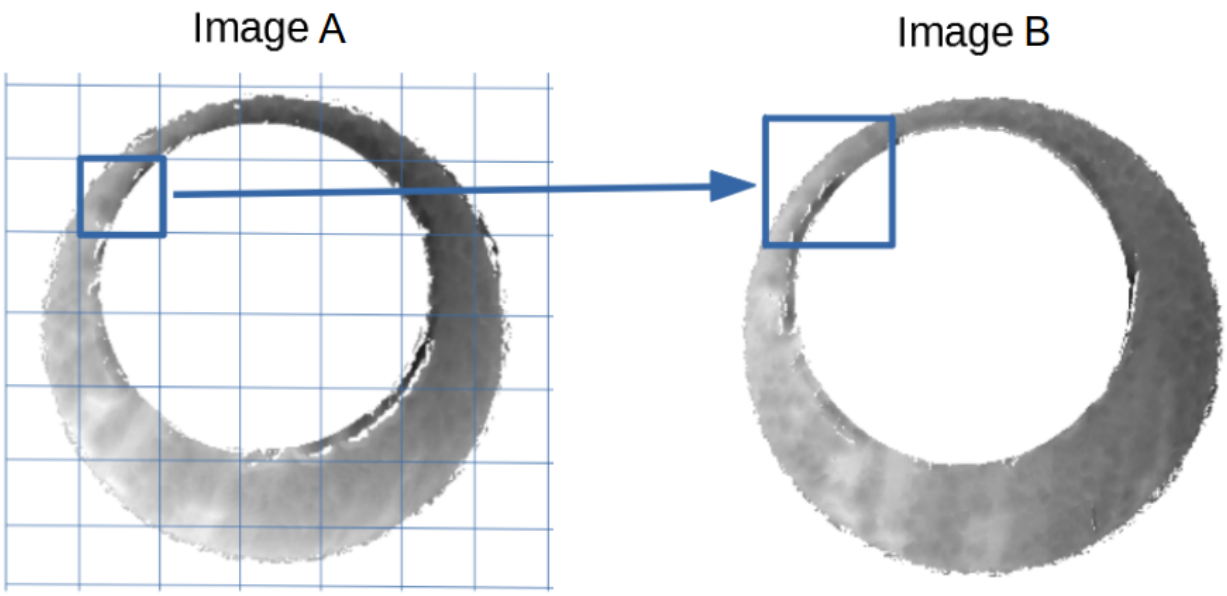


Figure 3.20: Illustration of comparing a "cell" in one cartrige case scan to a region in another.

Since the initial proposal of the CMC method, a number of improvements and extensions have been discussed by various authors (see section 2 for more information). The `cmcR` package provides an implementation for the initally proposed method as well as an "improved" algorithm proposed by Tong et al. (2015). The improved method builds upon the logic used by the initially proposed method, so both are explained below.

3.3.1 The Cross-Correlation Function

Similarity between two cells, say A_i and B_i, $i = 1, ..., n$, is measured using the maximum of the (normalized) cross-correlation function (CCF). For two real-valued, bivariate functions $f(t, s), g(t, s)$ where $t, s \in \mathbb{R}$, the *cross-correlation function*, denoted $(f \star g)(t, s)$, can be defined as

$$(f \star g)(t, s) := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau, \sigma) g(t + \tau, s + \sigma) \, d\tau d\sigma. \tag{3.1}$$

The integrand on the right hand side will be maximized when f, g are most "in-phase" with each other, i.e., are most similar. The (t, s) arguments allow us to compare the integral for different translated versions of g and determine for which translation values maximum similarity is achieved.

If f, g are of norm 1, then the CCF is bounded between -1 and 1 and can be interpreted similar to that of the correlation between two random variables. As such, it is common to first scale f, g to be of norm 1; in which case the CCF may be referred to as the *normalized cross-correlation function*. In practice, it is computationally infeasible to calculate the integral for every possible combination of (t, s). Instead, we can rely on the Cross-Correlation Theorem that says

$$\mathcal{F}(f \star g) = \overline{\mathcal{F}(f)} \cdot \mathcal{F}(g)$$

where $\mathcal{F}(\cdot)$ denotes the Fourier transform and $\overline{h}$ denotes the conjugate of a complex-valued function h. By applying an inverse Fourier transform of this equation, $\mathcal{F}^{-1}(\cdot)$, we get that $(f \star g) = \mathcal{F}^{-1}\left(\overline{\mathcal{F}(f)} \cdot \mathcal{F}(g)\right)$.

The cross-correlation function and Fourier transform extend to the discrete case. Suppose now that $A[m, n], B[m, n]$ are discrete (integer-valued), bivariate functions. For our purposes we can think of $[m, n] \in \mathbb{Z}^2$ as representing the location of a pixel in an image matrix and $A[m, n], B[m, n]$ as the pixel intensities at pixel $[m, n]$ of images A, B, respectively. The cross-correlation function in this case can be defined as

$$(A \star B)[m, n] := \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} A[i, j]B[m + i, n + j].$$

For a pair of images of size, say, $N \times N$, these sums would only need to be considered from $-N$ to N. For most images, it is not realistic to consider every possible combination of $[m, n]$. Instead, we can use both forward and inverse discrete fast Fourier transforms alongside the Cross-Correlation Theorem to obtain better computational efficiency. An estimate of the maximum value of the cross-correlation function will then be given by $\widehat{CCF}_{\max} = \max_{m,n} \mathcal{F}^{-1}\left(\overline{\mathcal{F}(A[m, n])} \cdot \mathcal{F}(B[m, n])\right)$ where $\mathcal{F}, \mathcal{F}^{-1}$ now denote the DFFT and IDFFT, respectively.

Note that the translation values at which $\widehat{CCF}_{\max}$ occurs are given by

$$[m, n] = \arg\max_{m,n} \mathcal{F}^{-1}\left(\overline{\mathcal{F}(A[m, n])} \cdot \mathcal{F}(B[m, n])\right).$$

We can clearly see that the CCF and convolution of two functions are highly similar to each other. The following example will build upon the example given in section 3.2.3.1 to see how the CCF and convolution differ.

Cross-Correlation Function Example In section 3.2.3.1 we considered a matrix created by taking a 100×100 matrix of 0s and adding to it a white (255 pixel value) 20×20 box in its center along with $N(0, 20)$ white noise. Now consider another matrix identical to the first, except that the box has now been shifted left 30 pixels and up 20 pixels. These matrices are shown in Figure 3.21.

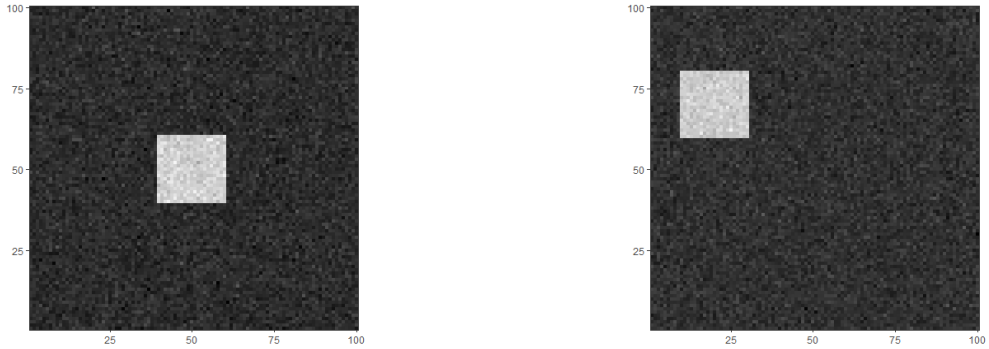


Figure 3.21: Two 100×100 matrices containing a 20×20 box and Gaussian white noise.

The CCF can be used to recover these translations and provide an interpretable quantity indicating how confident we can be that the recovered translations are correct. Figure 3.22 shows the cross-correlation function between the images in Figure 3.21. We see in particular that the CCF attains a maximum around $(30, -20)$, which is exactly the opposite of the translation performed on the second image. For this pair of images, the location at which the maximum CCF occurs can be interpreted as the translations needed to shift the second image to align with the first image. This indicates that, for any two images A and B, their CCF, $(A \star B)[m, n]$, can be interpreted from the "perspective" of image B. We also see that the CCF does not attain a maximum of 1; indicating that additional white noise added to the two images has, in some sense, muffled the signal of the white boxes.

3.3.2 The Initially Proposed Method

In their raw format, a pair of matching cartridge case scans are not necessarily properly aligned. If the surface matrices of two scans were stacked on top of one another such that their centers coincided, one would likely need to be rotated and translated to achieve maximum similarity with

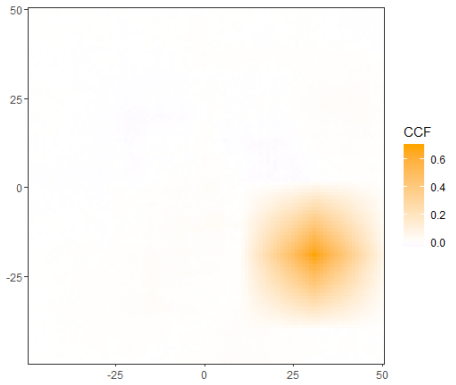


Figure 3.22: Cross-correlation function "map" between the two images in Figure 3.21.

the other. Once this is accomplished, an accurate similarity score can be calculated between the two scans. For two scans A and B, let $\boldsymbol{\beta} = (dx, dy, \theta)' \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ denote the translations and rotation needed to properly align A to B (so $-\boldsymbol{\beta}$ would align B to A). For the CMC method, we divide A into n cells and pair each cell with a region in B. As illustrated in Figure 3.20, the regions in B are defined based on the location of their associated cell in A, yet with larger side lengths.

Each cell will have an associated $\boldsymbol{\beta}_i = (dx_i, dy_i, \theta_i)', i = 1, ..., n$, by which it will achieve maximum similarity to the corresponding region in B. If A and B are truly matching (were fired from the same firearm), then we would expect $\boldsymbol{\beta}_i = \boldsymbol{\beta}$ for each i. Due to the unknowable nature of $\boldsymbol{\beta}$, the best we can hope to determine is if $\boldsymbol{\beta}_i = \boldsymbol{\beta}_j, i \neq j$. That is, one criterion specified to call a pair of cartridge case scans a match is that their cells' alignment parameters must agree. For non-matches, we would not expect the $\{\boldsymbol{\beta}_i\}_{i=1,...,n}$ to agree in this way but rather to differ randomly. Note that, as shown in Figure 3.20, some cells contain little to no observed values; for example, the corner cells of the Image 1 scan. Such cells are not considered when determining similarity, although some vaguity exists in how one might define a threshold to classify a particular cell as containing too few observed values. For example, Chen et al. (2017) only consider cells containing 15% or more observed values.

Because the cartridge cases are represented digitally, it is impossible to search over every possible value of $\boldsymbol{\beta}_i$. Instead, the alignment criterion, $\boldsymbol{\beta}_i = \boldsymbol{\beta}_j, i \neq j$, is applied to estimates for the alignment parameters, $\{\hat{\boldsymbol{\beta}}_i\}_{i=1,...,n}$. The translation parameter estimates are determined using the maximum of

the cross-correlation function (CCF). This is effectively a template-matching procedure, borrowing terminology from image processing, using the CCF as a similarity metric. The rotation parameter esimate is determined through a grid search. The method provides some estimation leeway by specifying a threshold, $\boldsymbol{T} = (T_{dx}, T_{dy}, T_\theta)'$, within which we say that two $\hat{\boldsymbol{\beta}}_i$s agree. The CCF is also used to quantify similarity between two cells analogous to how correlation can be used to measure the linear relationship between two random variables. In fact, a normalized CCF bounded between -1 and 1 is commonly considered for interpretability. An additional criterion to classify a cell pair as matching is a minimum CCF threshold, T_{CCF}. Note that these thresholds differ between authors.

In summary, the rotation parameter θ is estimated by comparing scan A to scan B over a grid of rotations of scan B. For each rotation of B, each cell in A is compared to its associated region in B using the CCF. This comparison yields both a maximum cross-correlation function value and an estimate of $(dx, dy)'$ for that particular rotation value. The method as initially proposed keeps track of the estimated alignment parameters by which each cell in scan A achieves its maximum CCF with the associated region in scan B. That is, each cell gets a single "vote" for the $\hat{\boldsymbol{\beta}}_i$ that provides the best alignment between it and its associated region in scan B. Cells that contain too few observed values aren't given a vote. A "consensus" is determined by aggregating in some way the $(dx, dy, \theta)'$ votes. For example, the median of the $(dx, dy, \theta)'$ estimates may be taken as the consensus. The votes of each cell are then compared to this consensus to determine if they are within the $(T_{dx}, T_{dy}, T_\theta)'$ thresholds. Any cell whose $\hat{\boldsymbol{\beta}}_i$ vote fall within these thresholds and whose maximum CCF is above the specified T_{CCF} threshold is classified as a "congruent matching cell."

3.3.3 The Improved Method

Tong et al. (2015) discuss an improved version of the CMC method that targets specific deficiencies in the initially proposed method. Namely, the fact that each image A cell gets only one vote for the $(dx, dy, \theta)'$ alignment parameters by which it achieves maximum similarity with its associated region in image B. A cell pair may be wrongfully excluded from the "congruent matching cell"

classification because, for example, the rotation for which it votes falls too far from the consensual rotation value. In reality, the cell pair may be highly similar at the consensual rotation value, yet is never given the chance to demonstrate this. The improved version of the method attempts to effectively give such cell pairs a second chance by considering their behavior at the consensual rotation value; although it introduces an additional criterion to guard against the possibility of a non-matching cell pair being highly similar by chance at the consensual rotation value.

This additional criterion is based on an observation that appears to be true based on empirical evidence: if a pair of cartridge case scans A and B are truly matching after rotating B by θ, then they should still be highly similar after rotating B by $\theta + \epsilon$ for some small ϵ. In terms of the CMC method, this means that the number of congruent matching cell pairs should reach a mode at the true θ value. In practice, because only a grid of θ values are considered, a mode is detected based on how close the θ values yielding a high number of CMCs are to each other. Tong et al. classifies a "high" number of CMCs as $\text{CMC}_{\text{max}} - \tau$ where CMC_{max} is the maximum number of CMCs detected across all θ values and τ is an empirical constant ($\tau = 1$ in Tong et al. (2015)).

Another improvement to the original method proposed by Tong et al. is to consider both a comparison of scan A to rotated versions of scan B as well as B to A. The motivation for this being that true matches should be highly similar in either direction while non-matches should have conflicting or random results. Figure 3.23 shows the "forward" and "backward" CMC count distribution for a known match pair across a grid of rotation values, $\theta \in \{-30, -27, ..., 27, 30\}$. We can see that both distributions achieve a mode at opposite θ values, which is expected and desired for a known match pair. Additionally, a horizontal line has been drawn at $\text{CMC}_{\text{high}} = \text{CMC}_{\text{max}} - 1$ for the respective distributions. We can see that in either case, any θ value that achieves this CMC_{high} value is close to the mode. In contrast, Figure 3.24 shows the CMC count distribution for a known non-match pair that is relatively flat. In particular, we see that there are θ values achieving CMC_{high} that are far away from the mode.

These additional "high CMC count" criteria provide more enhanced discriminatory ability between known matching and known non-matching scans. If this criterion is satisfied by a particular

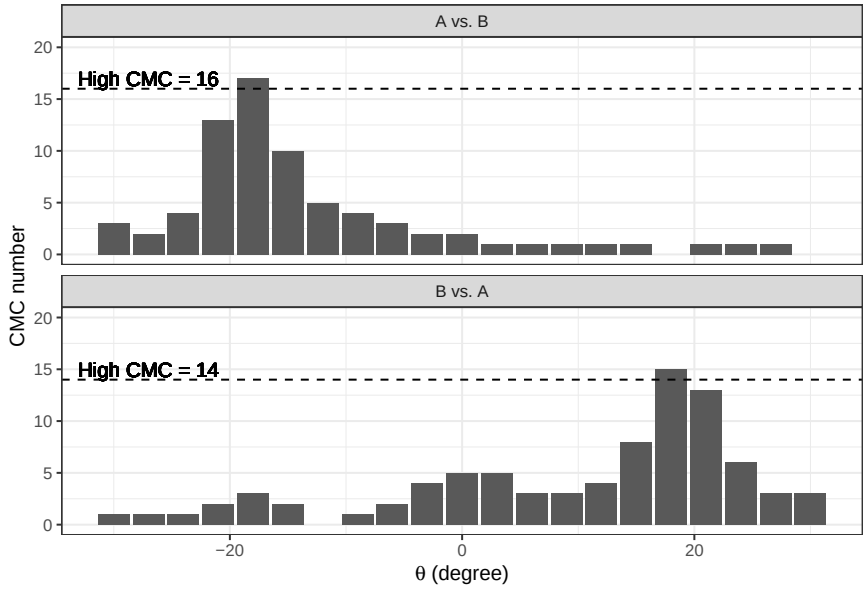


Figure 3.23: CMC count per rotation (θ) value for Forward (A vs. B) and Backward (B vs. A) comparison of a known match cartridge case scan pair

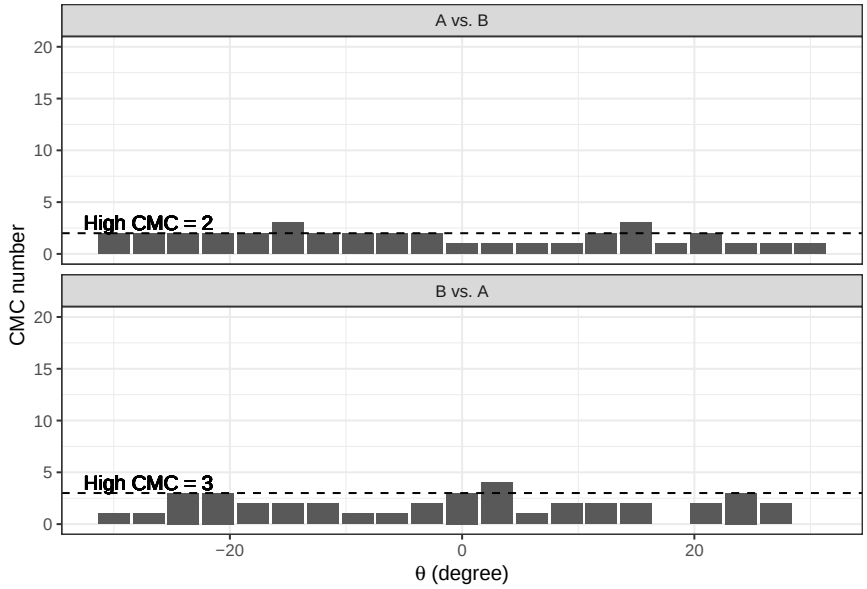


Figure 3.24: CMC count per rotation (θ) value for Forward (A vs. B) and Backward (B vs. A) comparison of a known non-match cartridge case scan pair

pair of scans and a θ mode is identified, then the method considers CCF values for each cell pair at and around this mode. Again, this is in contrast to the initially proposed method that only considers the $\mathrm{CCF}_{\max}$ value and resulting (dx, dy, θ) votes for each cell pair.

CHAPTER 4. cmcR PACKAGE

This chapter will highlight the `cmcR` package's functionality by walking through a possible use case. Many of the functions in this package provide the user with a variety of processing options. This is motivated from the fact that processing techniques differ considerably among authors and appear to lead to different results. The specifics of the techniques used are often missing in the literature, for example how surface matrices are resized to some nominal dimension, so the authors of this package have decided to provide a non-exhaustive list of options which with users can experiment.

The cartridge cases used in this example originate from a study done by Fadul et al. (2011). They are openly accessible through the NIST Ballistics and Tools Marks Research Database (specifically, here).

4.1 Pre-processing procedures

Studies in which cartridge cases are matched by forensic examiners often involve giving examiners a set of known matches and asking them to classify additional matches from a collection of unknown source scans (see Baldwin et al. (2014) for an example). Consider such a situation in which we are given four cartridge case scans; two of which are known matches and two of which are of unknown source. We wish to determine whether either of the unknown source scans match the two known source scans.

The four cartridge cases' surface matrices prior to pre-processing are represented in Figure 4.1. The top two cartridge cases are the known matches, so we'll refer to as Known Source 1 and Known Source 2 from left to right. The bottom two cartridge cases we'll refer to as Unknown Source 1 and Unknown Source 2 from left to right.

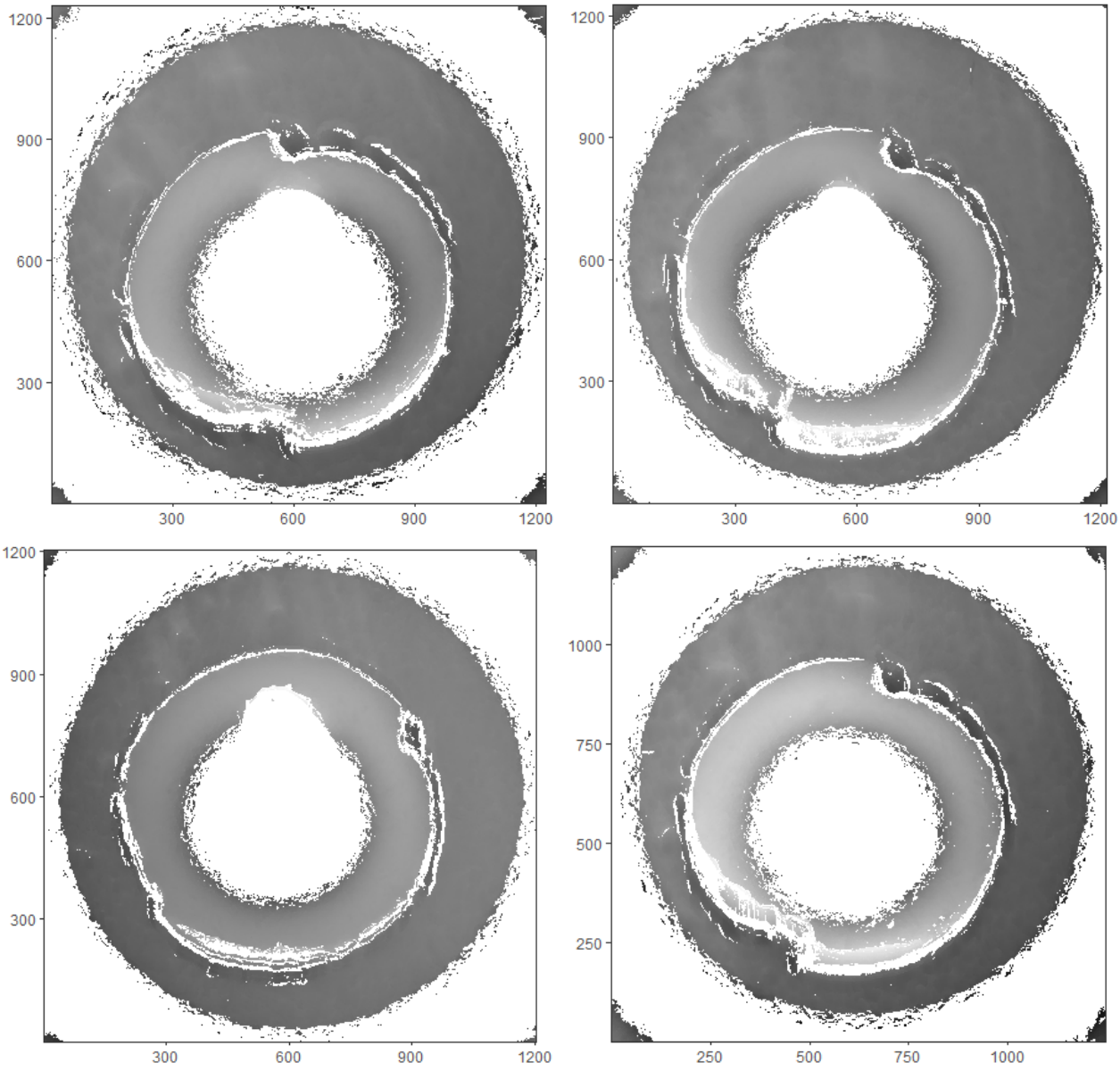


Figure 4.1: (Top) Two known matching scans. (Bottom) Two scans of unknown source.

The pre-processing steps detailed in section 3.2 are implemented in the `cmcR` package via the family of functions beginning with `preProcess_`. To recap, these include the following:

1. Determine the height value of the breech face impression region in the scan. This is accomplished using Random Sample Consensus (RANSAC) that determines a plane of best fit out of a number of candidate planes fit to the scan. The plane with the most number of "inliers," defined to be points within a set distance of the fitted plane, is chosen. The `preProcess_ransac` function accomplishes this. The number of candidate planes to be fit and the inlier distance threshold can be specified by the user through the `ransacIters` and `ransacInlierThresh` arguments, respectively. See section 3.2.1.1 for more information.

2. Extract the breech face impression region from the scan by taking all observed values within the inlier threshold set for the RANSAC method. The resulting matrix is then "centered" by subtracting from each element the matrix's average value. The `preProcess_levelBF` function accomplishes this. It is also possible to take the residuals between the fitted plane and observed values by setting the `useResiduals` argument to `TRUE`. See section 3.2.1.1 for more information.

3. Crop-out unnecessary columns/rows of `NA`s on the exterior of the breech face impression region. This is done by considering the middle 20% of rows/columns in the scan (since the corners typically contain observed values) and finding the first and last of these rows/columns to contain a certain number of observed values. The `preProcess_cropWS` function accomplishes this. The `croppingThreshold` argument controls the minimum number of observed values that need to be observed in a particular row/column for it to not be cropped. See section 3.2.1.2 for more information.

4. Identify the firing pin impression circle in the center of the scan and filter out any pixels within this circle. Identification is performed first by estimating the radius of the firing pin impression circle and then using a Hough transform to detect the center of this circle in

the scan. The `preProcess_removeFPCircle` accomplishes this. See section 3.2.1.3 for more information.

For resizing surface matrices, two exterior functions have commonly been used. The `x3ptools::sample_x3p` function will, as the name suggests, sample every `m`th element of the surface matrix, starting from the bottom left corner. The `imager::resize` function uses various interpolation methods to resize an image to an arbitrary size. See section 3.2.2 for more information on these various resizing techniques.

Some authors discuss using a Gaussian filter to highlight the breech face impression "signal" in the scan. For example, Tong et al. (2014) use a low-pass Gaussian filter on the cartridge case scans with standard deviation equal to 1. The `cmcR::preProcess_gaussianFilter` function implements either a low-pass Gaussian filter, used to reduce the effects of high-frequency noise in the scan, a high-pass Gaussian filter, used to reduce the effects of low-frequency global structure, or a Gaussian bandpass filter, which performs both a low and high-pass Gaussian filter. See section 3.2.3.1 for more details on how Gaussian filtering is implemented for image processing.

Considering our current use case, the following shows a possible set of pre-processing steps that could be used on one of the scans.

```
x3p1 <- x3ptools::read_x3p("path/to/file.x3p") %>%
  x3ptools::sample_x3p(m = 2,offset = 0)


x3p1$surface.matrix <- x3p1$surface.matrix %>%
  preProcess_ransac(inlierTreshold = 10^(-5), #1 micron
                    finalSelectionThreshold = 2*10^(-5), #2 microns
                    iters =  200) %>%
  preProcess_levelBF(useResiduals = TRUE) %>%
  preProcess_cropWS(croppingThresh = 2) %>%
  preProcess_removeFPCircle() %>%
  preProcess_gaussFilter(res = x3p$header.info$incrementY,
```

```
    wavelength = 16,
    filtertype = "lp") #lowpass filter
```

Alternative to these pipe-friendly, modularized pre-processing functions, one can accomplish the same steps with one function call using the `selectBFImpression` function. See the cmcR documentation for more information about this all-in-one alternative. The four processed scans are shown in Figure 4.2. Note that the matrices' dimensions have been reduced due to downsizing and cropping out whitespace on the exterior of the scans. Also, markings on these processed scans are more visually obvious than on the original scans only because the images' grayscale pixel values are scaled based on the surface matrices' height values. The original scans contain more observations of variable height than these processed scans making smaller changes in height less visually obvious.

4.2 Calculating the cross-correlation function

Once the scans are processed, we can calculate the cross-correlation function for each cell pair across different of rotations as outlined in section 3.3. That is, we can divide the cartridge cases into cell pairs and calculate the cross-correlation function for each cell pair for various rotations of one of the images. The `cmcR::cellCCF` function is the main driver of this procedure. It accepts as arguments two x3p objects (`x3p1` and `x3p2`, a vector of rotation values in degrees by which the surface matrix stored in `x3p2` is to be rotated (`thetas`), the number of cells that the surface matrices should be split into (`cellNumHoriz` and `cellNumVert`, and the minimum proportion of a cell that needs contain observed (non-NA) values (`minObservedProp`). Optionally, each cell can be centered by the average value in the overall surface matrix or per cell by setting `centerCell = "wholeMatrix"` or `centerCell = "individualCell"`, respectively. Similarly, `scaleCell = "wholeMatrix"` and `scaleCell = "individualCell"` scales a cell by the standard deviation of the entire matrix or of itself, respectively. Ott et al. (2017) discuss using the "areal cross correlation function" defined for two matrices A and B of size $M \times N$ to be

$$ACCF = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{A_{ij} - \mu_A}{\sigma_A} \frac{B_{ij} - \mu_B}{\sigma_B}$$

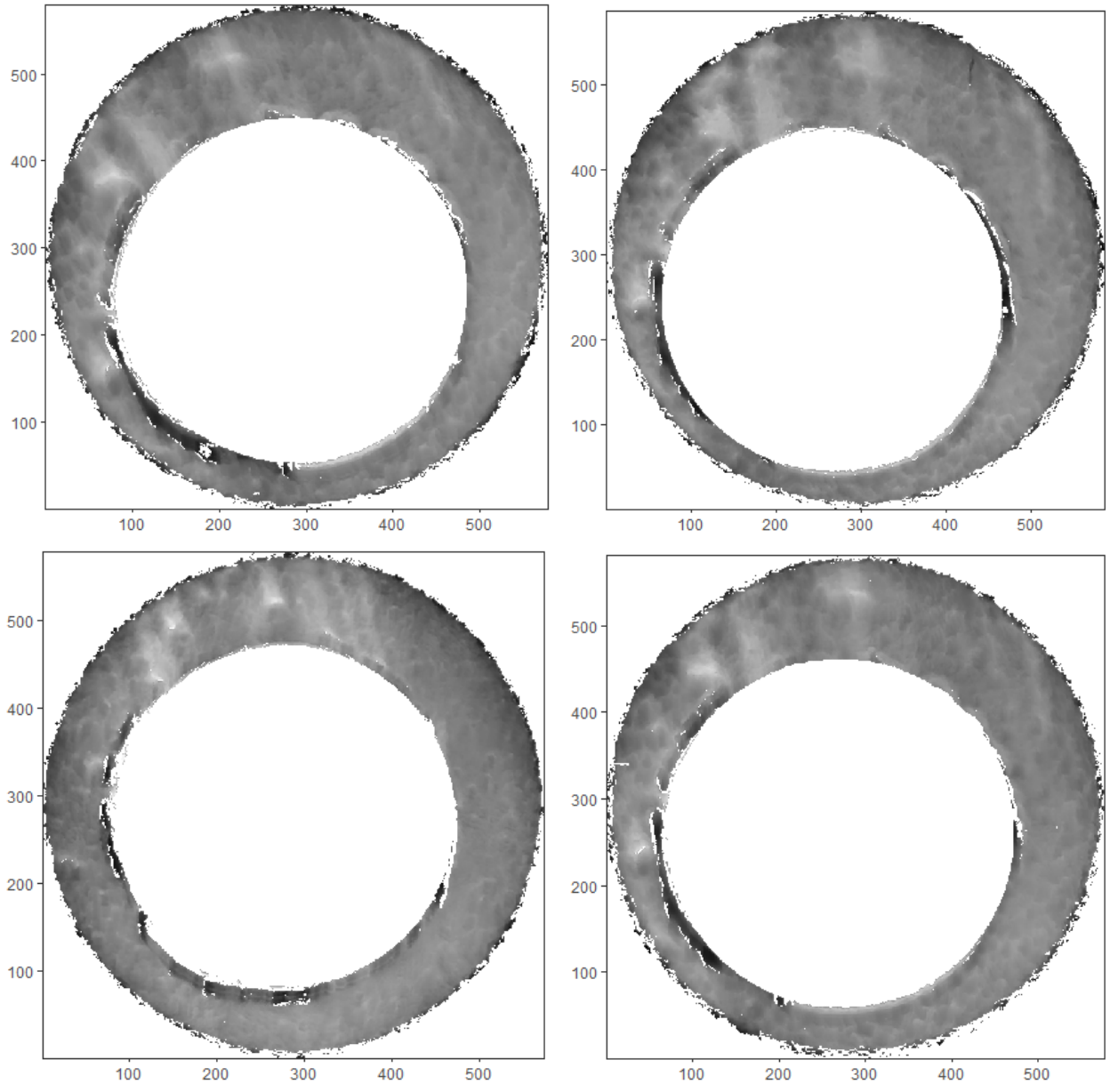


Figure 4.2: (Top) Two processed scans of known source. (Bottom) Processed scan of unknown source.

where μ_A, μ_B are the average average values of the elements of A, B, respectively, and σ_A, σ_B are the standard deviations of A, B, respectively. This seems to indicate that it is standard to center/scale each matrix before computing the CCF. By default, the `centerCell` and `scaleCell` arguments are left empty, in which case centering/scaling does not occur.

The call to `cellCCF` below will compare two surface matrices separated into an 7×7 array of cells over a grid of rotations of the second surface matrix from -30 to 30 degrees. Note that this particular grid of rotations is used by Tong et al. (2015). Only cells containing 15% or more observed values are considered for CCF calculations. Each cell is centered by the average value in the overall matrix.

```
comparison1 <- cmcR::cellCCF(x3p1 = processedBF1$x3p,
                             x3p2 = processedBF2$x3p,
                             thetas = seq(-30,30,by = 3),
                             cellNumHoriz = 7,
                             cellNumVert = 7,
                             regionToCellProp = 4,
                             minObservedProp = .15,
                             centerCell = "individualCell",
                             scaleCell = "individualCell")
```

The value returned by `cellCCF` is a list of two elements. The first element, `params`, contains the argument specifications under which the function call was made. The second element, `ccfResults`, is a list of data frames, one per rotation value, of the CCF results per cell. Each cell pair is identified with a `cellID` string corresponding to a range of rows and columns in the surface matrix of `x3p1` by which the cell is defined. For example, a cell ID of `y = 1 - 144, x = 289 - 432` corresponds to a cell consisting of rows 1-144 and columns 289-432 of the surface matrix. The $(dx, dy)'$ values associated with the maximum CCF value are interpreted to be the translations, assuming the cell and region pairs are stacked such that their centers coincide, needed to shift the `x3p2` regions to achieve maximum CCF with the `x3p1` cells.

4.3 Congruent Matching Cells logic

With the CCF results calculated across a range of rotation values, we can extract the features used in the Congruent Matching Cells method.

The `cmcR::topResultsPerCell` function extracts from the list returned by `cellCCF` information about the $(\theta, dx, dy)'$ values at which each cell pair achieves its maximum CCF. The resulting data frame is required for determining the number of CMCs under the initially proposed method. Recall that the initially propsed method filters this data frame down by determining which cell pairs have associated $(\theta, dx, dy)'$ values close to the consensus $(\theta, dx, dy)'$. Passing this data frame to the `cmcR::cmcFilter` function implements this filtering process. The `cmcFilter` function provides options to specify the function used to form a consensus among $(\theta, dx, dy)'$ values and define thresholds for calling a particular cell pair's $(\theta, dx, dy)'$ values "close" to the consensus. Optionally, separate consensus functions can be defined for the θ and $(dx, dy)'$ values. This is motivated by the fact that, especially for a course grid, the θ values can be considered categorical while the $(dx, dy)'$, although only ever observed as integers, should be treated as continuous. The code below illustrates how CMCs under the initially proposed method can be calculated. The `cmcR::getMode` function calculates the mode θ value among those in the `topResultsPerCell` data frame.

```
comparison1$ccfResults %>%
  topResultsPerCell() %>%
  cmcFilter(consensus_function = median,
            ccf_thresh = .4,
            dx_thresh = 15,
            dy_thresh = 15,
            theta_thresh = 3,
            consensus_function_theta = median)
```

Tables 4.1 and 4.2 show the CMCs calculated under the initially proposed method for the comparisons between Unknown Source 1 and Known Source 1 and 2, respectively. In contrast,

tables 4.3 and 4.4 show the initial CMCs for the comparisons between Unknown Source 2 and Known Sources 1 and 2. There is clearly better evidence to suggest similarity between Unknown Source 2 and the known source cases than Unknown Source 1. Song (2013) proposes using 6 CMCs as a threshold to declare a cartridge case pair a "match," although experimentation. It is indeed the case that Unknown Source 1 is a non-match to both cartridge cases of known source while Unknown Source 2 is a match.

cellNum	cellID	ccf	dx	dy	θ
24	y = 146 - 217, x = 503 - 573	0.5	2	8	6

Table 4.1: 1 initial CMC for a comparison between Known Source 1 and Unknown Source 1.

cellNum	cellID	ccf	dx	dy	θ
40	y = 290 - 361, x = 503 - 573	0.52	-9	-2	9

Table 4.2: 1 initial CMC for a comparison between Known Source 2 and Unknown Source 1.

cellNum	cellID	ccf	dx	dy	θ
5	y = 1 - 73, x = 289 - 360	0.56	-1	3	18
15	y = 74 - 145, x = 433 - 504	0.46	11	13	21
39	y = 291 - 362, x = 433 - 504	0.43	-1	-1	18
46	y = 363 - 435, x = 361 - 432	0.45	1	0	18
50	y = 436 - 507, x = 73 - 144	0.58	-6	-7	21
59	y = 508 - 579, x = 145 - 216	0.41	1	0	18

Table 4.3: 6 initial CMCs for a comparison between Known Source 1 and Unknown Source 2.

The initial CMCs between Unknown Source 2 and the two known source scans are shown in Figure 4.3. The `cmcR::cmcPlot` function can create such plots given a "questioned" cartridge case and the associated CMC data frame like the one returned by `cmcFilter`.

While useful for an initial estimate of the CMC count between two cartridge case scans, the improved method as described in section 3.3.3 has been shown to be more effective at differentiating matches from non-matches. The `cmcR::cellCCF_bothDirections` is just a wrapper for calling `cellCCF` twice where the two x3p objects change roles for the second call. The value returned by `cellCCF_bothDirections` is a list of two elements, the first being the comparisons between `x3p1`

cellNum	cellID	ccf	dx	dy	θ
4	y = 1 - 73, x = 220 - 292	0.51	6	-15	6
5	y = 1 - 73, x = 293 - 365	0.50	4	-14	6
14	y = 74 - 146, x = 366 - 438	0.44	0	-13	3
46	y = 366 - 438, x = 366 - 438	0.50	3	-17	3

Table 4.4: 4 initial CMCs for a comparison between Known Source 2 and Unknown Source 2.

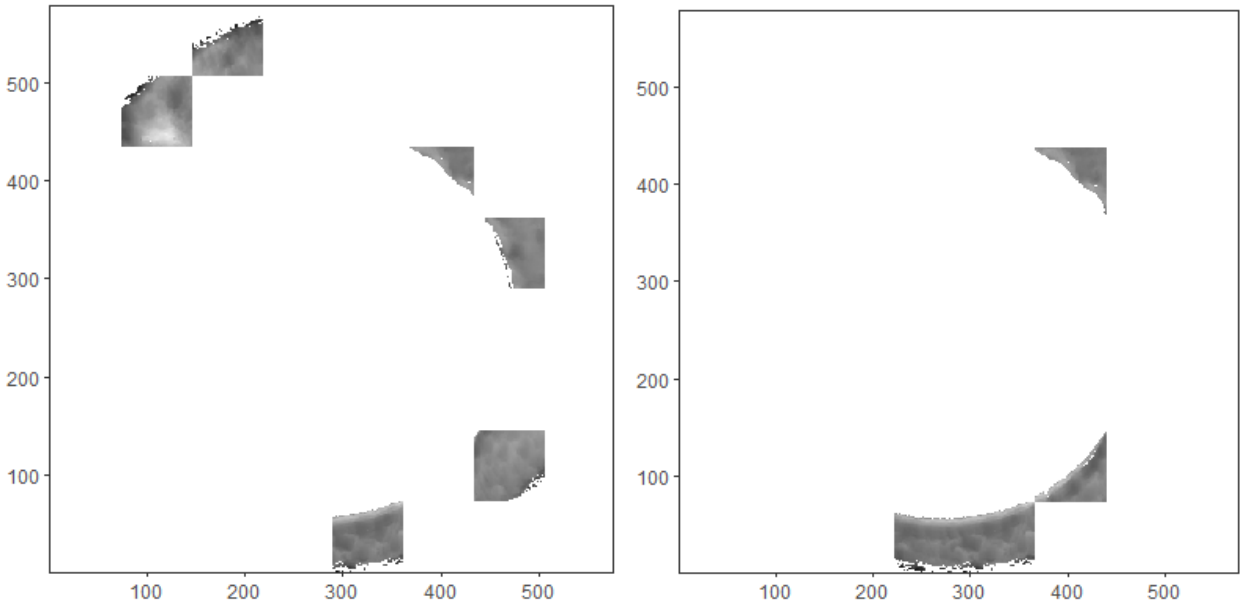


Figure 4.3: Initial CMCs for the comparison between Unknown Source 2 and (left) Known Source 1 and (right) Known Source 2.

to rotations of `x3p2` and the second being the comparisons between `x3p2` to rotations of `x3p1`. The output of `cellCCF_bothDirections` can be handed off to the `cmcR::cmcFilter_improved` function to implement the improved CMC method. The value returned by `cmcR::cmcFilter_improved` is a list of 3 elements. The first element, `params`, contains the argument specifications under which the function call was made. The second element, `initialCMCs`, contains two dataframes of the CMCs calculated under the initially proposed method (equivalent to calling `cmcFilter` twice). The third element, `finalCMCs`, contains a data frame with the CMCs calculated under the improved method.

```
comparison2 <- cellCCF_bothDirections(processedBF1,
                                      processedBF2,
                                      thetas = seq(-30,30,by = 3),
                                      cellNumHoriz = 8,
                                      cellNumVert = 8,
                                      regionToCellProp = 4,
                                      minObservedProp = .15,
                                      centerCell = "individualCell",
                                      scaleCell = "individualCell")


cmcR::cmcFilter_improved(cellCCF_bothDirections_output = comparison2,
                         consensus_function = median,
                         ccf_thresh = .4,
                         dx_thresh = 15,
                         dy_thresh = 15,
                         theta_thresh = 3)
```

If no additional CMCs are determined using the improved method, then a cartridge case pair is assigned the number of CMCs calculated under the initially proposed method. Such is the case for the comparison between Unknown Source 1 and Known Source 1. Tables 4.5 and 4.6 show the final CMCs calculated under the improved method for the comparison between Unknown Source 2 and

the two known source scans. The column "comparison" corresponds to the fact that the improved method requires comparing a single cartridge case pair twice - where both scans play the role of the "questioned" scan to which the other is compared.

cellNum	cellID	ccf	dx	dy	θ	comparison
3	y = 1 - 73, x = 146 - 217	0.52	13	-10	-15	comparison_1to2
4	y = 1 - 73, x = 217 - 288	0.50	-2	2	21	comparison_2to1
5	y = 1 - 73, x = 289 - 360	0.56	-1	3	18	comparison_2to1
10	y = 74 - 145, x = 73 - 144	0.58	7	-3	21	comparison_2to1
14	y = 74 - 145, x = 362 - 433	0.43	7	6	-21	comparison_1to2
15	y = 74 - 145, x = 433 - 504	0.46	0	3	18	comparison_2to1
17	y = 146 - 218, x = 1 - 72	0.44	-1	6	21	comparison_2to1
18	y = 146 - 218, x = 74 - 145	0.46	-6	6	-21	comparison_1to2
23	y = 146 - 218, x = 434 - 505	0.42	-1	-3	-18	comparison_1to2
24	y = 146 - 218, x = 506 - 577	0.44	1	1	-15	comparison_1to2
25	y = 219 - 290, x = 1 - 72	0.64	-1	4	18	comparison_2to1
31	y = 219 - 290, x = 433 - 504	0.44	-5	8	15	comparison_2to1
33	y = 291 - 362, x = 1 - 72	0.45	0	5	18	comparison_2to1
39	y = 291 - 362, x = 434 - 505	0.53	-1	-1	-21	comparison_1to2
41	y = 363 - 435, x = 1 - 73	0.56	-10	-12	-15	comparison_1to2
42	y = 363 - 435, x = 74 - 145	0.50	7	6	-21	comparison_1to2
46	y = 363 - 435, x = 361 - 432	0.45	1	0	18	comparison_2to1
48	y = 363 - 435, x = 505 - 576	0.50	4	-10	21	comparison_2to1
50	y = 436 - 507, x = 73 - 144	0.58	-6	-7	21	comparison_2to1
53	y = 436 - 507, x = 290 - 361	0.51	8	-5	-21	comparison_1to2
55	y = 436 - 507, x = 434 - 505	0.46	0	-1	-18	comparison_1to2
59	y = 508 - 579, x = 146 - 217	0.55	-6	0	-18	comparison_1to2
60	y = 508 - 579, x = 217 - 288	0.61	10	1	15	comparison_2to1
61	y = 508 - 579, x = 289 - 360	0.55	7	-2	18	comparison_2to1
62	y = 508 - 579, x = 362 - 433	0.42	-9	4	-15	comparison_1to2

Table 4.5: 25 final CMCs for a comparison between Known Source 1 and Unknown Source 2.

The final CMCs for Unknown Source 2 are shown in Figure 4.4.

cellNum	cellID	ccf	dx	dy	θ	comparison
4	y = 1 - 73, x = 220 - 292	0.51	6	-15	6	comparison_1to2
5	y = 1 - 73, x = 289 - 360	0.51	-10	15	-6	comparison_2to1
10	y = 74 - 145, x = 73 - 144	0.56	1	5	-3	comparison_2to1
14	y = 74 - 145, x = 361 - 432	0.59	-1	15	-3	comparison_2to1
17	y = 147 - 219, x = 1 - 73	0.56	2	-13	6	comparison_1to2
23	y = 147 - 219, x = 439 - 511	0.44	4	-13	9	comparison_1to2
25	y = 219 - 290, x = 1 - 72	0.68	-6	19	-6	comparison_2to1
31	y = 219 - 290, x = 433 - 504	0.69	0	14	0	comparison_2to1
32	y = 220 - 292, x = 512 - 583	0.46	8	-7	6	comparison_1to2
33	y = 291 - 362, x = 1 - 72	0.44	-5	16	-6	comparison_2to1
40	y = 293 - 365, x = 512 - 583	0.43	11	-10	9	comparison_1to2
41	y = 363 - 435, x = 1 - 72	0.47	-10	-11	0	comparison_2to1
42	y = 366 - 438, x = 74 - 146	0.42	-3	-19	6	comparison_1to2
46	y = 366 - 438, x = 366 - 438	0.50	3	-17	3	comparison_1to2
48	y = 366 - 438, x = 512 - 583	0.51	13	-12	3	comparison_1to2
50	y = 439 - 511, x = 74 - 146	0.57	-5	-6	6	comparison_1to2
51	y = 439 - 511, x = 147 - 219	0.44	20	-3	0	comparison_1to2
60	y = 508 - 579, x = 217 - 288	0.51	3	-1	-6	comparison_2to1
61	y = 512 - 583, x = 293 - 365	0.56	18	-1	0	comparison_1to2

Table 4.6: 19 final CMCs for a comparison between Known Source 2 and Unknown Source 2.

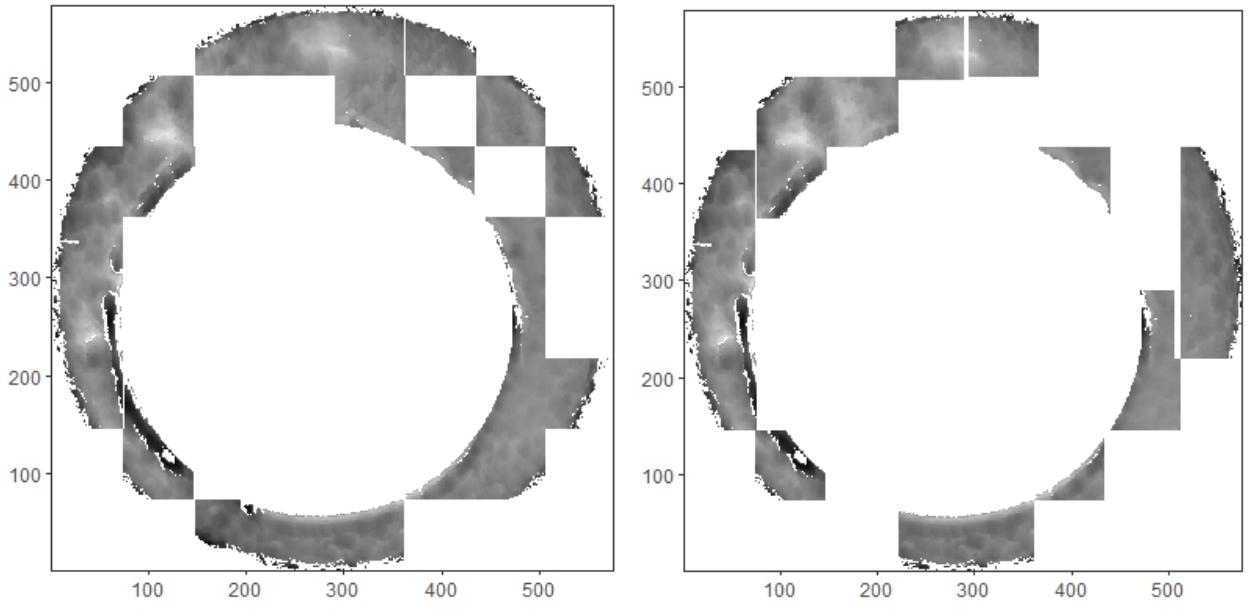


Figure 4.4: Final CMCs for the comparison between Unknown Source 2 and (left) Known Source 1 and (right) Known Source 2.

CHAPTER 5. RESULTS AND DISCUSSION

This chapter will focus on results analogous to those commonly given in the CMC-based literature. A popular set of test case cartridge cases are those from Fadul et al. (2011). The `cmcR` package has been applied to these cartridge cases under a wide variety of processing conditions. Notable results will be shared here. As alluded to in previous chapters, the final results of the Congruent Matching Cells method seem to be highly dependent on the myriad pre, inter, and post-processing decisions that need to be made for a given comparison. Authors provide different levels of detail of what went into a particular comparison. Because of this, a total of 8 pre-processing different pre-processing conditions have been considered.

A useful tool for determining how effective a particular procedure is at differentiating matches from non-matches is the distribution of CCF_{max} scores for each known match and known non-match pair. A necessary assumption in calculating CCF_{max} using the Cross-Correlation theorem (i.e., the FFT-based method) is that neither matrix being compared contains missing (`NA` in our case) values. However, the processed cartridge case scans contain many deliberately placed missing values (e.g., pixels within firing pin circle). As such, the FFT-based method for calculating CCF_{max} requires replacing these missing values with something. The `cmcR` package uses the convention of first subtracting away from all non-missing elements of the matrix the average value in the matrix and then replacing any missing values with 0. Zero-imputing a signal in this fashion means that the CCF_{max} values obtained by applying the Cross-Correlation theorem don't accurately reflect the similarity between a cell pair. The left side of Figure 5.1 shows the distribution of FFT-based CCF_{max} values obtained for all known match and known non-match cells of the Fadul et al. (2011) cartridge cases. Specifically, these values are based on taking the raw values from the RANSAC method and not applying a Gaussian filter. There is a large amount of overlap between the distribution of scores for known match and known non-match cells. The counts given in this

plot correspond to the number of cell pairs of each type represented in the plot. While the CCF_{max} scores obtained in this way aren't reliable measures of similarity, the $(dx, dy)'$ translation values at which the CCF_{max} values occur are very often reliable estimates of the true translation values at which a cell pair is most similar. We can use these translation values to "stamp-out" from the larger region a matrix of the same dimension as the smaller cell. Both of these matrices will contain missing values as opposed to replacing them with 0. The "pairwise-complete" correlation, which is a correlation computed only from pairs of non-missing elements, can be computed for these matrices. The right side of Figure 5.1 shows the CCF_{max} values computed using this pairwise-complete procedure. We can clearly see that the CCF_{max} values are considerably higher for both known match and known non-match cell pairs. There is still an issue of considerable overlap, but this may indicate that taking the raw values from the RANSAC method and not applying some sort of Gaussian filter may not be the most effective pre-processing conditions.

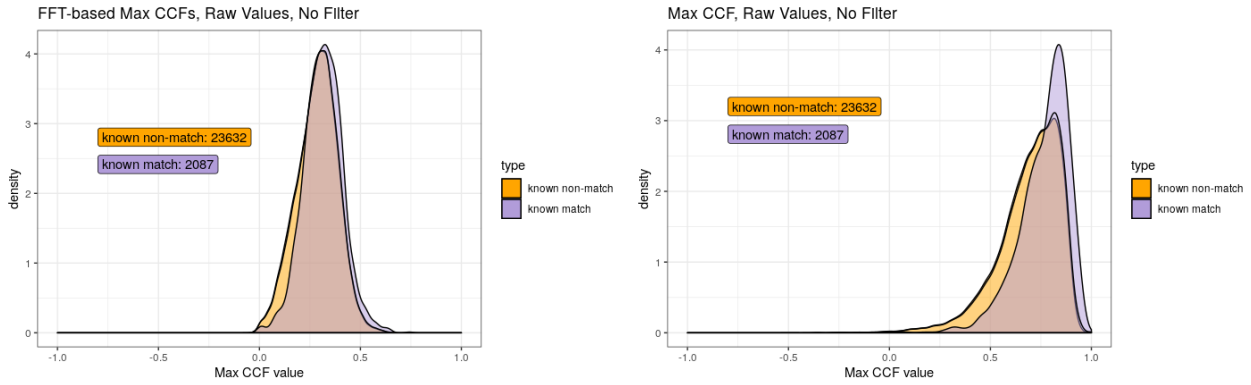


Figure 5.1: CCF_{max} distribution as calculated using the Cross-Correlation theorem vs. pairwise-complete observations.

Figure 5.3 shows the CCF_{max} distributions under a variety of pre-processing conditions. The left column contains results if the "raw" values close to the RANSAC-fitted plane are taken while the right column contains results if the residuals between the plane and the observed values are taken. The counts provided in each plot are the number of known match and known non-match cell pairs represented by the densities. These counts differ between each plot because the RANSAC method used to detect the breech face impression height value is based on randomly sampling points within

the scan. This inherent randomness yields slightly different fitted planes from one use to the next. This could cause, for example, a cell that would be included in one comparison to no longer be included in another. In the current version of the `cmcR` method, it is suggested to run the RANSAC method twice, once with a inlier threshold of 1 micron (1×10^{-6} meters) and use this output to run the method again with an inlier threshold of .1 microns. LP, HP, and BP stand for low, high, and bandpass Gaussian filters applied to each surface matrix prior to comparison. Tong et al. (2014) uses a lowpass filter with a wavelength cutoff of approximately 30 microns (specifically, a Gaussian kernel with standard deviation $\sigma = 1$ on matrices of size 700×700). As discussed previously, Chen et al. (2017) use a second-order robust Gaussian regression filter with wavelength cutoffs of 16 and 250 microns. The current version of the `cmcR` package does not implement a Gaussian regression filter, but taking the residuals from the RANSAC method and applying a bandpass filter performs a similar procedure.

We can see from Figure 5.3 that there is some evidence to suggest that CCF values based on the residuals are more "well-behaved" than those of the raw values. This may be due to some large-scale structure in a particular cartridge case, for example, a "tilt" in the scan due to inherent manufacturing specifications or due to the scanning process, that isn't removed when considering the raw values. Figure 5.2 shows two processed known match cartridge case scans in which the raw values were taken from the RANSAC method rather than the residuals. Recall that height values are mapped to grayscale pixel intensities in these plots. The fact that one side of these scans is darker than the other indicates that there is a global trend to the scan. It is our goal to match two cartridge cases based on the breech face impressions left be a firearm, not based on artifacts of the manufacturing process or because two scans happened to be taken under similar conditions. Two cartridge cases of the same brand/manufacturer may be fired by two completely different firearms, so including such information in a scan may actually lead to an increase in false positives.

The purpose of using a highpass Gaussian filter is to reduce the effects of such global, "large wavelength" structure in the scans seen in Figure 5.2. This explains why highpass filtering the raw

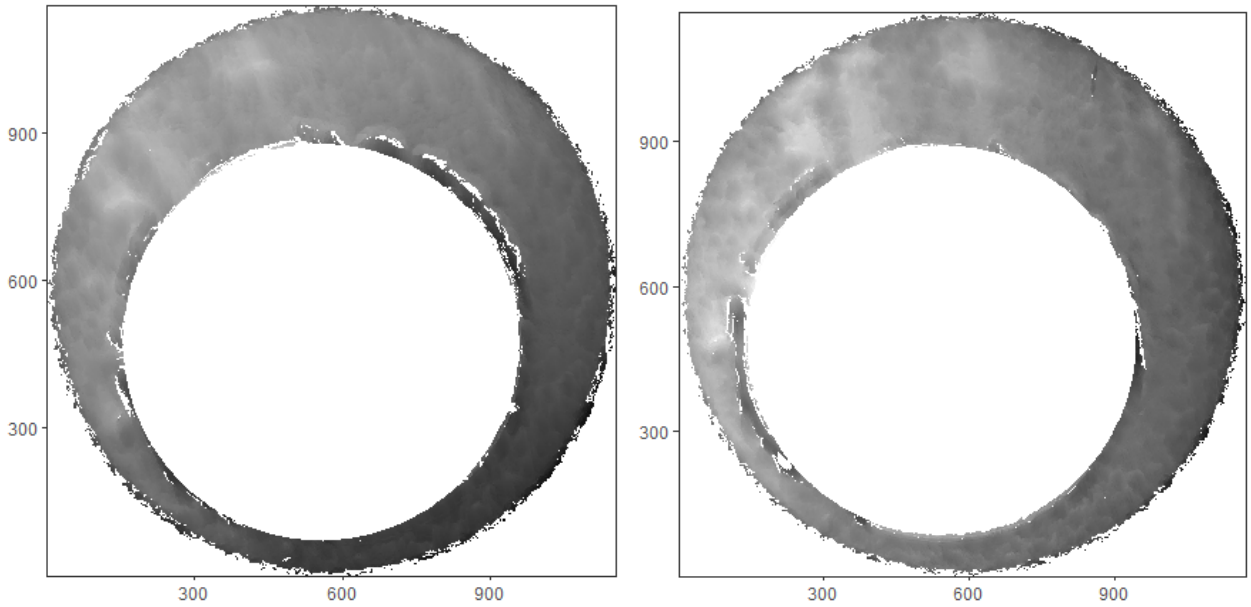


Figure 5.2: The "raw" values returned from the RANSAC method for a known match pair.

or residual values returned from the RANSAC method yields better separated known match vs. known non-match CCF_{max} values.

As discussed in that section, a cell pair is declared a "match" if its associated dx, dy, θ and CCF_{max} values pass certain thresholds related to their proximity to consensus-based dx, dy, θ values. It is important to determine how sensitive the initially proposed and improved classification methods are to such thresholds. For example, Figure 5.4 shows the CMC distributions under the initially proposed method for the bandpass filtered, residual-valued cartridge case pairs for various CCF_{max} and translation classification thresholds. From left to right, the CCF_{max} minimum threshold stays constant while the translation dx, dy thresholds become less restrictive (10 units away from the median vs. 15 units ... vs. 30 units). From top to bottom, the CCF_{max} minimum threshold becomes more restrictive. Thus, the most restrictive of these faceted plots is in the bottom-left corner. Such plots are useful for determining how sensitive the CMC distribution is to the correlation and translation thresholds. In terms of actual classification accuracy, Figure 5.9 shows the Receiver Operating Characteristic (ROC) curves associated with each CMC distribution in Figure 5.5 by changing the minimum CMC count threshold used to classify a match (Bradley

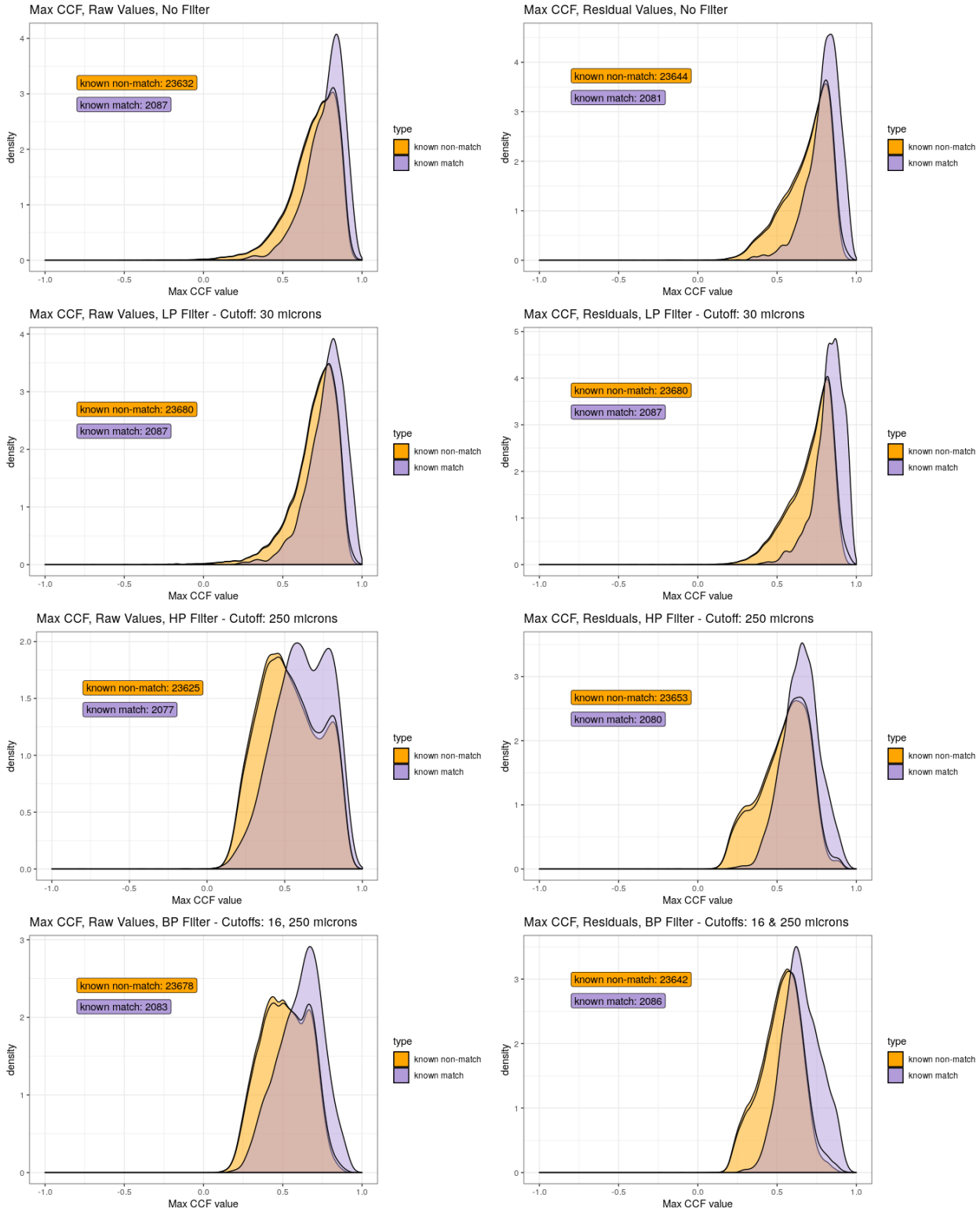


Figure 5.3: $\mathrm{CCF}_{\mathrm{max}}$ distribution under various pre-processing conditions.

(1997)). We can see that a number of these ROC curves have associated Area Under the Curve (AUC) close to 1, indicating that near-complete separation of the known match and known non-match CMC distributions is attained.

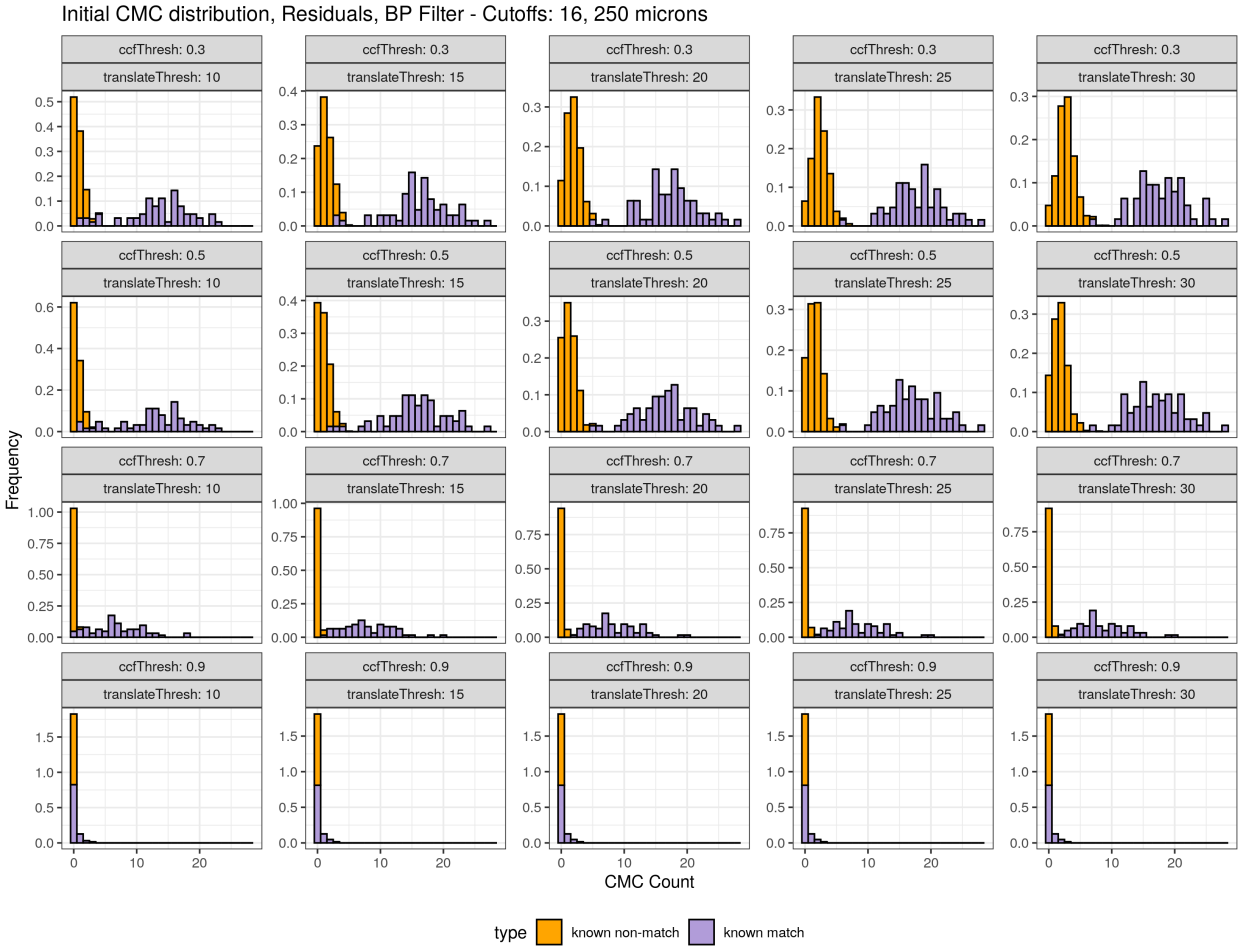


Figure 5.4: Initial CMC distributions for various filtering thresholds on bandpass filtered, residual-valued known match and known non-match pairs

To demonstrate that perfect classification is attainable with the initially proposed method, Figure 5.6 shows the initial CMC distributions for the bandpass filtered, residual-valued known match and known non-match pairs under a finer grid of CCF and translation thresholds. An AUC value of 1 indicates that there is a complete separation of the known match and known non-match initial CMC distributions.

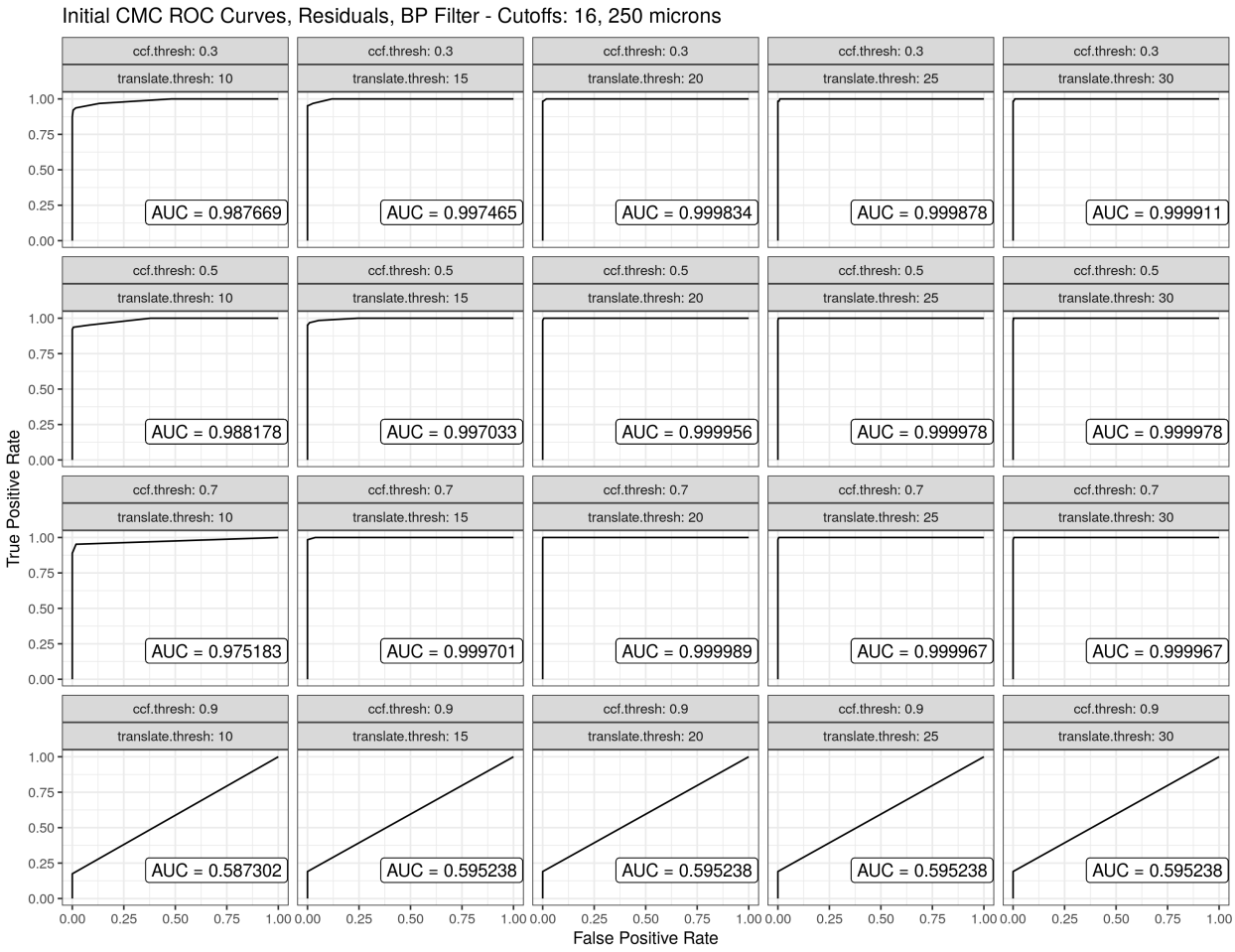


Figure 5.5: ROC curves associated with the CMC distributions of Figure 5.4 by varying the CMC count classification threshold

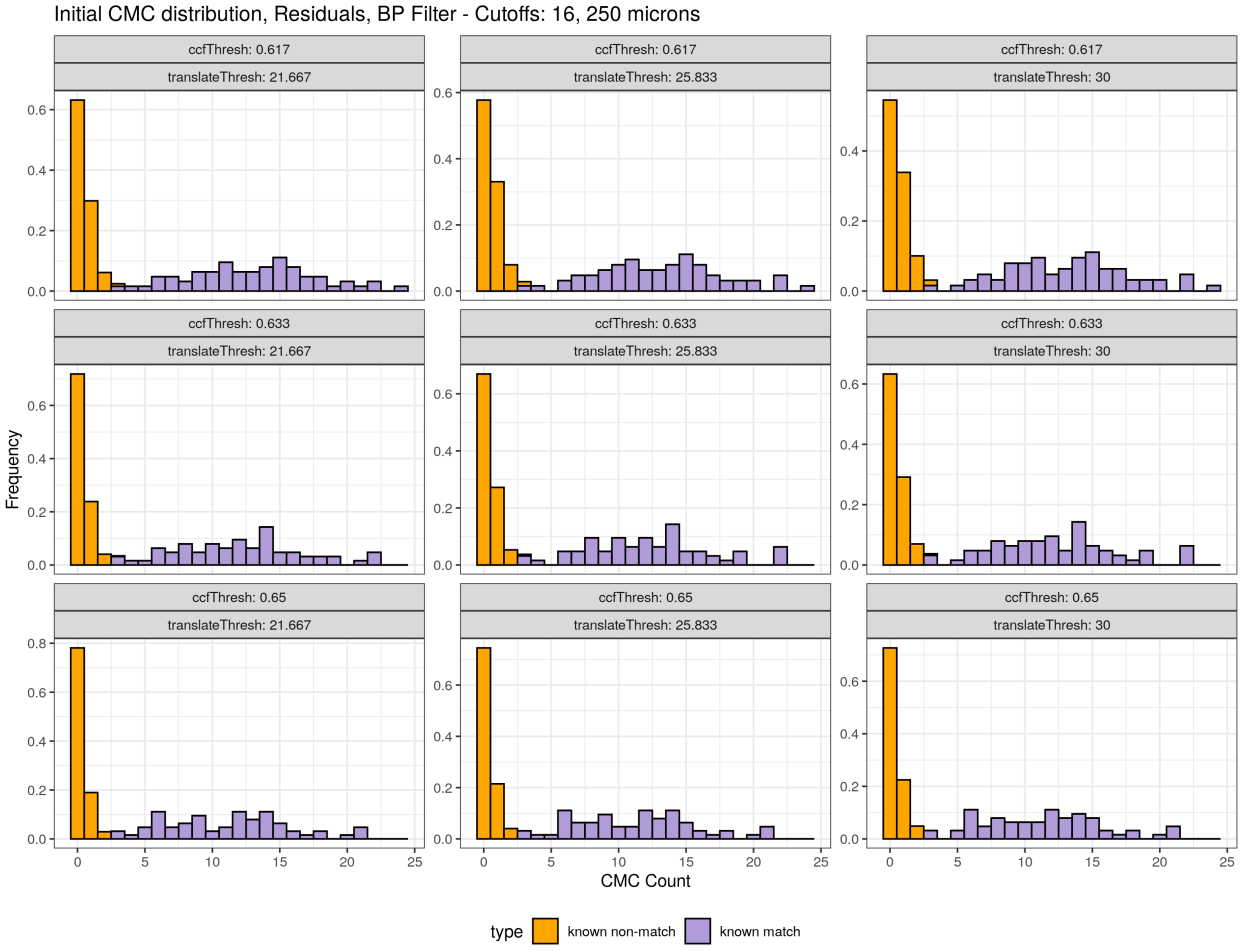


Figure 5.6: Initial CMC distributions for a finer grid of CCF_{max} and translation thresholds on bandpass filtered, residual-valued known match and known non-match pairs

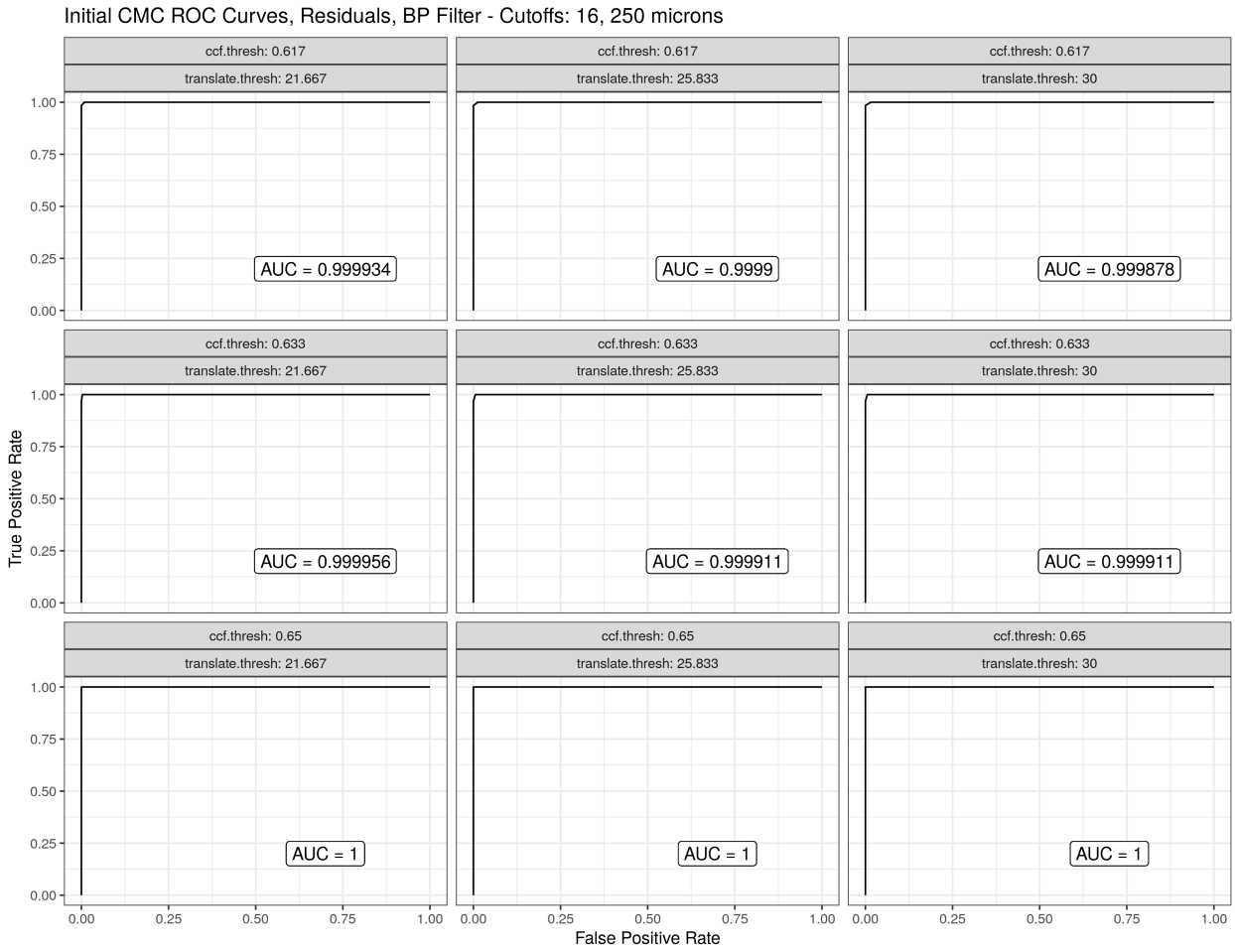


Figure 5.7: ROC curves associated with the initial CMC distributions of Figure 5.6 by varying the CMC count classification threshold

Figure 5.8 shows the "final" CMC distributions using the improved CMC method logic under various threshold values for the bandpass filtered, residual-valued cartridge case pairs. Similar to the initial CMC distributions' ROC curves, a number of these ROC curves have associated Area Under the Curve (AUC) close to 1, indicating that near-complete separation of the known match and known non-match CMC distributions is attained. Figure 5.10 shows an alternative view of these ROC plots by faceting by the CCF threshold and coloring by the translation threshold. We can better see the within-CCF relationship between the various translation thresholds. In particular, we can see how larger translation thresholds (which is more flexible than smaller) tend to permit higher false positive classifications, although this behavior is really only visible in the top two ROC plots.

To demonstrate that perfect classification is attainable with the improved method, Figure 5.11 shows the final CMC distributions for the bandpass filtered, residual-valued known match and known non-match pairs under a finer grid of CCF thresholds. Figure 5.12 shows their associated ROC curves by varying the CMC count classification threshold. A number of these plots show an AUC of 1, indicating that perfect separation is achieved between the known match and known non-match CMC distributions. Interestingly, from a classification perspective, the initially proposed and improved method yield comparable results based on these conditions. In particular, we can see that known non-matches are more often assigned high CMC values than in the initially proposed method. We can see in many of the plots, namely those in the top row of Figure 5.8, that the known non-match CMC distributions are bimodal. This can likely be attributed to some known non-match pairs passing the additional criteria applied in the improved method. Any known non-match pairs that do "slip underneath" these criteria are assigned an inflated number of CMCs relative to those known non-match that don't. It is common in CMC-related literature to compare two methods based on how separated the known match and known non-match CMC distributions are from each other (namely, how far the largest known non-match CMC count is from the smallest known match CMC count). By this metric, it should be noted that many authors have been able to achieve better separation between known match and known non-match CMC distributions. Undoubtedly,

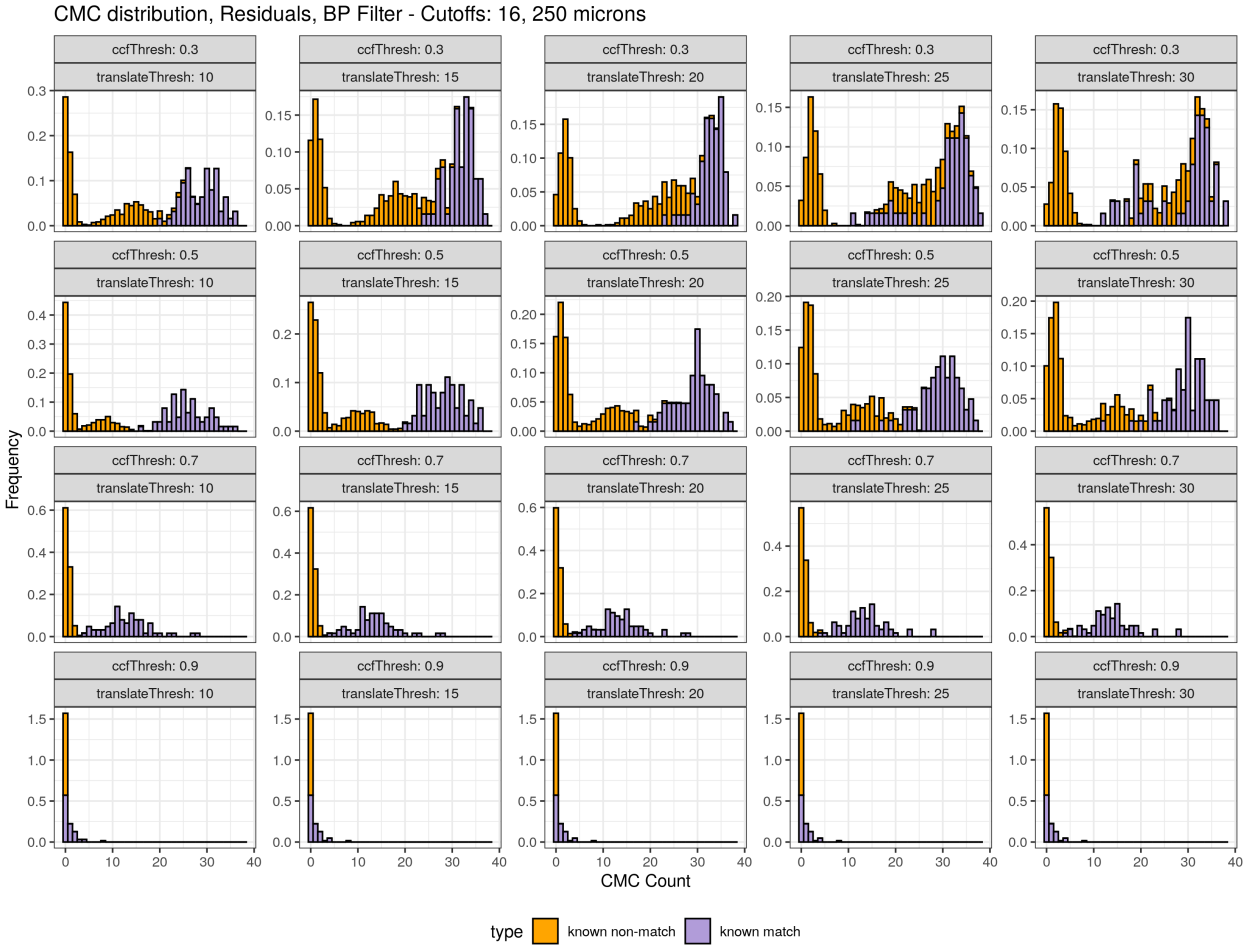


Figure 5.8: Final CMC distributions for various filtering thresholds on bandpass filtered, residual-valued known match and known non-match pairs

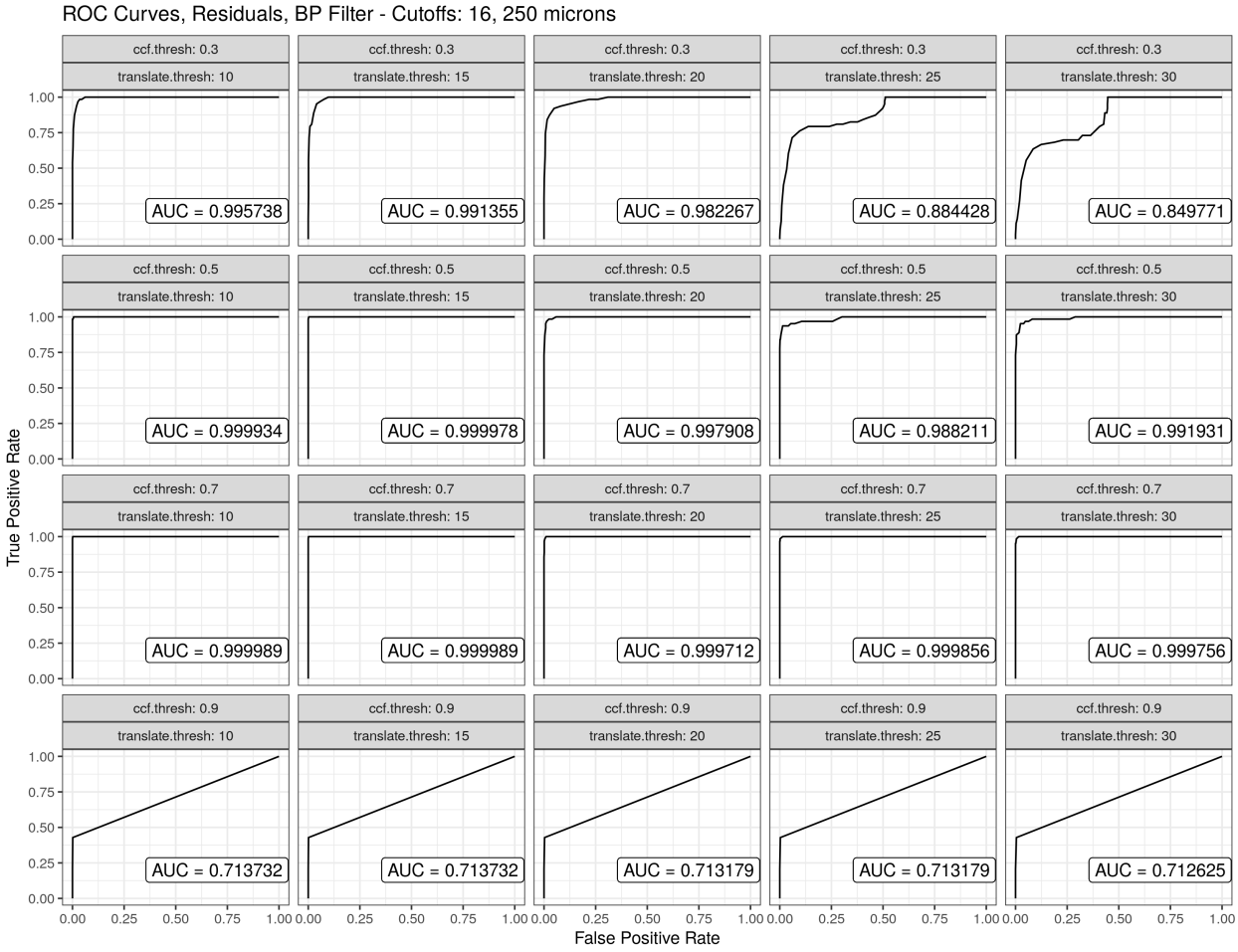


Figure 5.9: ROC curves associated with the final CMC distributions of Figure 5.8 by varying the CMC count classification threshold

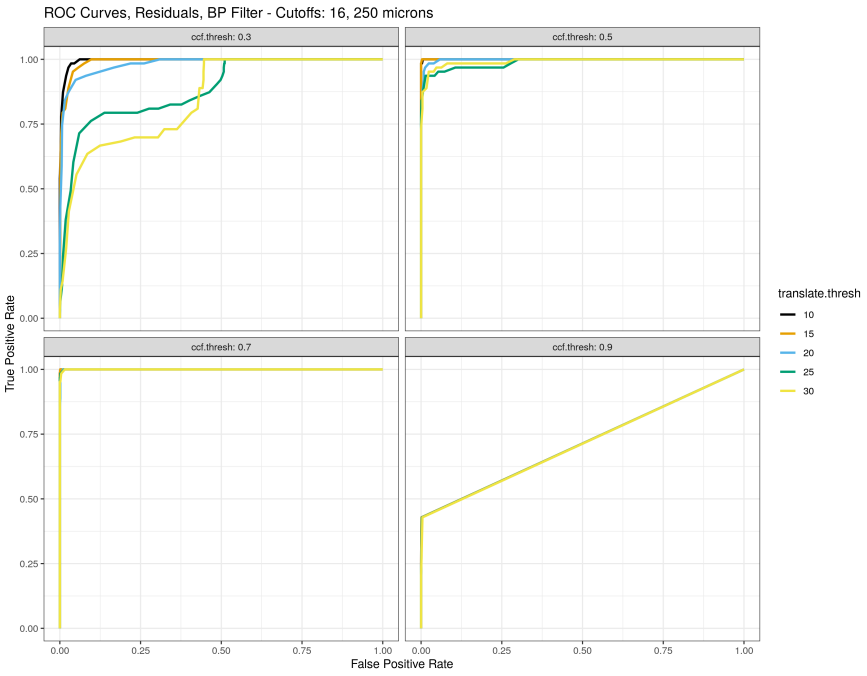


Figure 5.10: ROC curves faceted by the CCF threshold associated with the final CMC distributions of Figure 5.8 by varying the CMC count classification threshold

tuning the pre, inter, and post-processing conditions used would yield better results. Additionally, minor tweaks to the implementation of the CMC logic could certainly affect the issues posed here.

See Appendix A for the CMC distributions and associated ROC curves for the 7 other pre-processing conditions whose CCF_{max} distributions are represented in Figure 5.3.

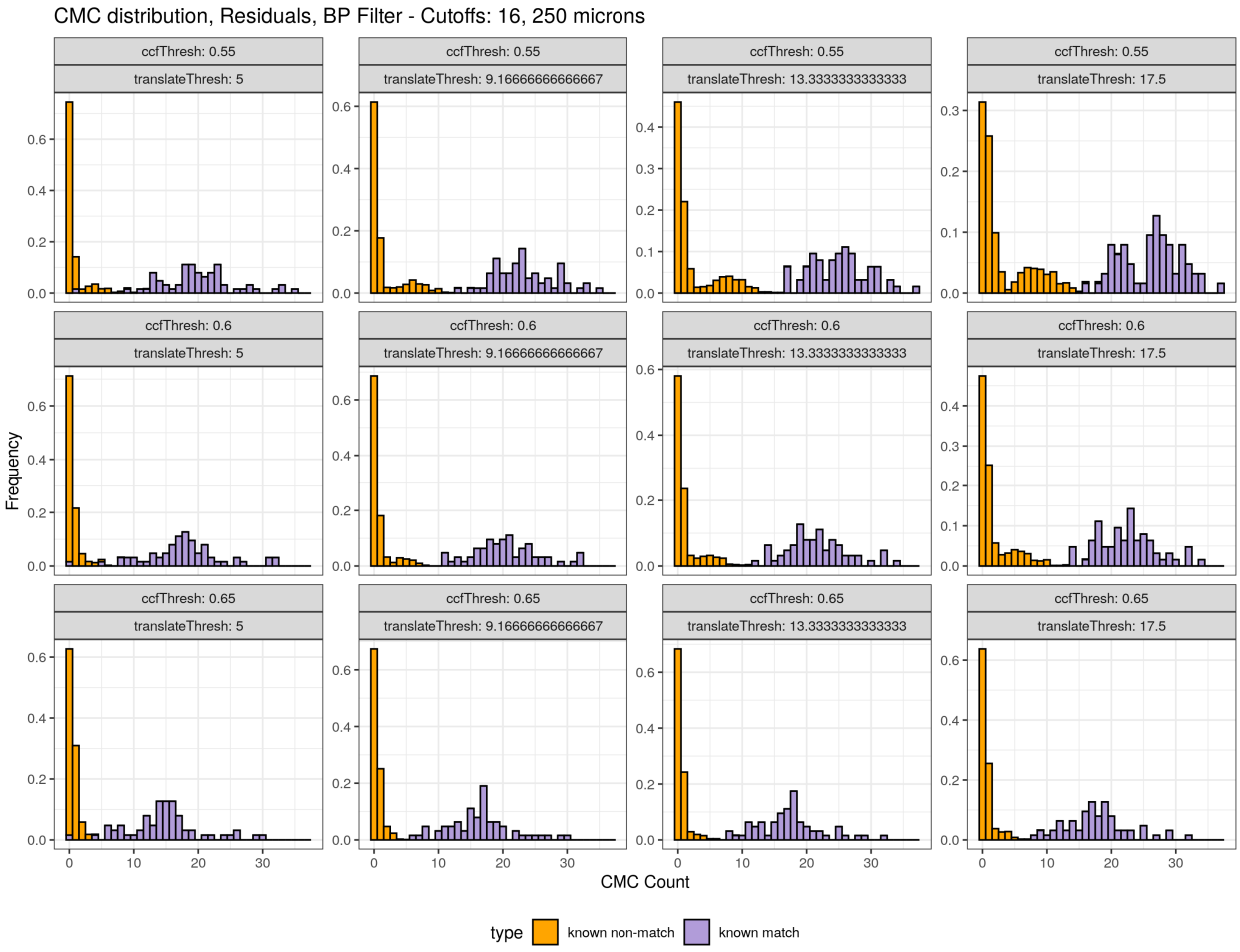


Figure 5.11: Final CMC distributions for a finer grid of CCF_{max} and translation thresholds on bandpass filtered, residual-valued known match and known non-match pairs

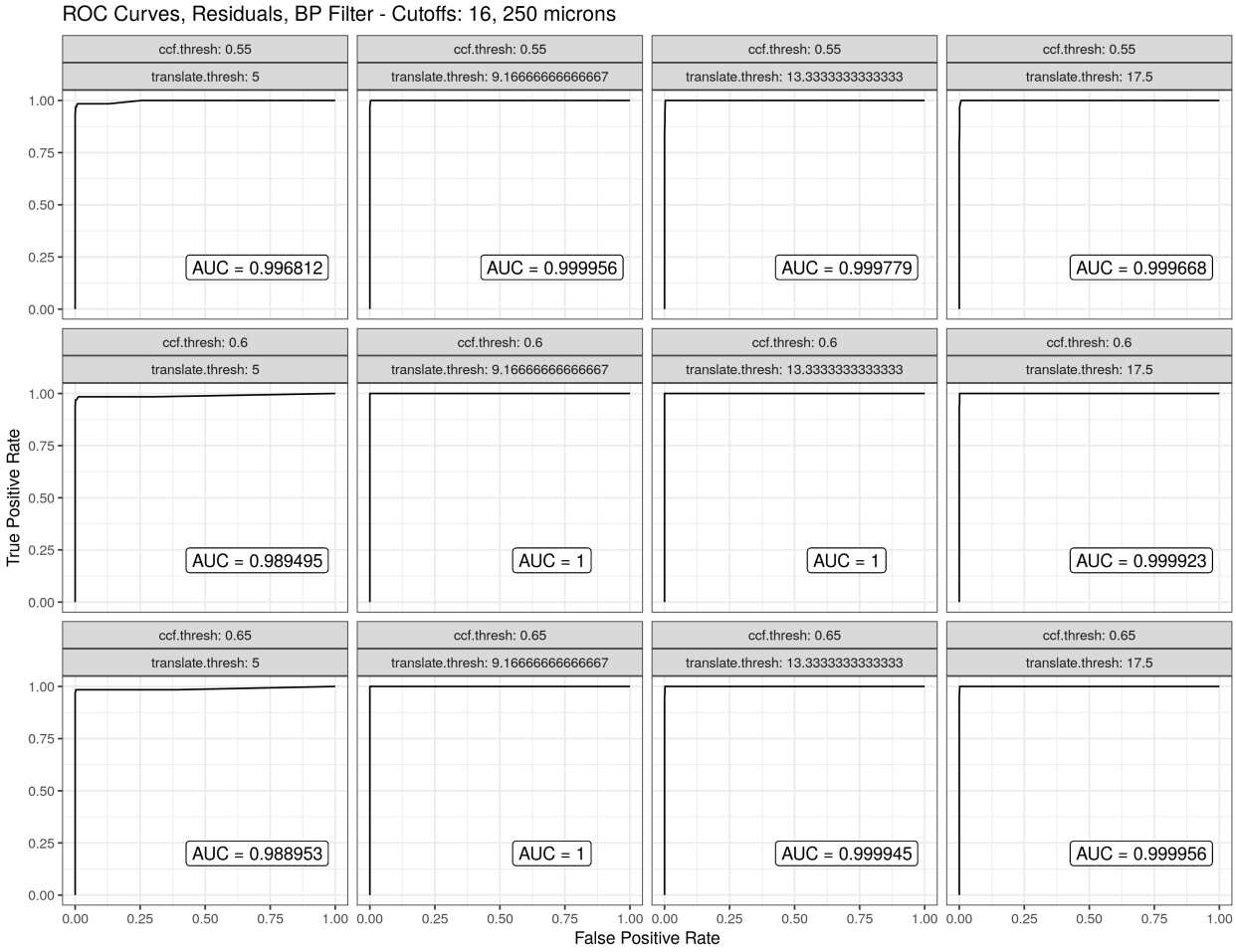


Figure 5.12: ROC curves associated with the final CMC distributions of Figure 5.11 by varying the CMC count classification threshold

CHAPTER 6. CONCLUSION AND FUTURE WORK

The results presented in this manuscript indicate that the open-source implementation of the CMC method provided in the `cmcR` package is a close replication to the implementations discussed in Song (2013) and Tong et al. (2015). In particular, as discussed in Chapter 5, we can achieve perfect classification of matches and non-matches under a number of thresholding parameter combinations. This indicates some level of robustness to the method's ability to separate matches from non-matches under different conditions. It should be noted that authors such as Tong et al. (2015) and Chen et al. (2017) were able to achieve better separation of the known match and known non-match CMC distributions than what has currently been achieved using the `cmcR` package. However, further tuning of the pre, inter, and post-processing parameters would undoubtedly yield results even closer to those reported in other CMC-related literature.

Another likely contributor to these differing results is the exact implementation of the various procedures used in the CMC method. A function like `fft` may be implemented differently in R than it is in, say, MATLAB. Assuming that both implementations are valid, even small numerical differences could compound to yield largely different results, especially with how often the `fft` functions is used in the calculation of the cross-correlation function. This is why it is so important to elucidate black-box methods by providing open-source implementations. The implementation of some pre-processing procedures discussed in the CMC literature, such as the second-order robust Gaussian regression filter used by Chen et al. (2017). Such a "regression filter" is effectively a locally-weighted regression with weights determined by a Gaussian density centered at the point to be estimated. Making a regression filter "robust" involves penalizing any estimates that are too far from the majority of other estimates. Muralikrishnan and Raja (2008) provide details of how to construct robust second-order regression filters in 2D as well as how to construct (non-robust) zero-order Gaussian regression filters in 3D; the difference in orders being whether a constant func-

tion or a quadratic is fit locally to a surface. Dr. Muralakrishnan is currently a mechanical engineer in the Dimensional Metrology Group in the Sensor Science Division of the Physical Measurement Laboratory at NIST, so presumably the implementation of this second-order robust Gaussian regression filter used in Chen et al. (2017) is from in-house (and thus not openly available). It would be useful to implement such a method in the future.

A popular way to compare various CMC methods is to determine how effective each are at separating the known match and known non-match CMC distributions (i.e., how far the maximum CMC count is for known non-matches from the minimum CMC count for known matches). However, it may also be informative to compare methods based on the associated spread of the CMC distributions. For example, Fung (1995) discusses two diagnostic statistics for discriminant analysis that could be applied to these CMC distributions. Additionally, as discussed in Chapter 2, many recent CMC-related publications have focused on various ways to deal with missing values in the cartridge case scans. As discussed in Chapter 5, using the FFT-based method to calculate the CCF_{max} for a particular cell/region pair, while significantly faster than a "brute-force" search over every possible translation of one matrix relative to another, does not yield reliable CCF_{max} estimates. Using the "pairwise-complete" observations to calculate the CCF is similar to the "valid data-based normalized cross-correlation" method proposed by Tong et al. (2018). However, there are certainly a number of other techniques that could work to handle missing values. For example, angle-preserving transformations may make cartridge case data more amenable to cell-based classification methods.

As discussed in chapter 5, the most promising results, and the authors' current recommendations, are based on the following conditions, which are also the default parameter values in the `cmcR` package implementation. For a particular pair of cartridge case scans:

1. Down-sample the cartridge case scans by a factor of 2 as described in section 3.2.2.

2. Estimate the breech face impression height value in the scan using the RANSAC method. Numerically stable results have been found by applying the method twice: once with a relatively large inlier threshold (e.g., 1 micron) and again with a smaller threshold (e.g., .1 microns).

3. Crop-out whitespace on the exterior of the cartridge case scan in the surface matrix

4. Use a Hough transform method to detect and remove any pixels inside of the firing pin impression circle

5. Apply a Gaussian bandpass filter with wavelength cut-offs of 16 and 250 microns

6. Perform a cell-based comparison method by partitioning one scan into a grid of 8×8 cells. Scale and shift all cells/regions considered in the comparison by their respective mean and standard deviation, respectively. Consider comparisons over a grid of rotations $\theta \in \{-30, -27, ..., 30\}$.

7. Apply the "improved" CMC method to the resulting comparison data using the median as the consensus function, a CCF_{max} threshold of .6, a translation threshold of 10 units, and a rotation threshold of 3 degrees.

BIBLIOGRAPHY

Baldwin, D. P., Bajic, S. J., Morris, M., and Zamzow, D. (2014). A Study of False-Positive and False-Negative Error Rates in Cartridge Case Comparisons:. Technical report, Defense Technical Information Center, Fort Belvoir, VA.

Barthelme, S. (2019). *imager: Image Processing Library Based on 'CImg'*. R package version 0.41.2.

Bourke, P. (1998). Image filtering in the frequency domain.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698.

Chen, Z., Song, J., Chu, W., Soons, J. A., and Zhao, X. (2017). A convergence algorithm for correlation of breech face images based on the congruent matching cells (CMC) method. *Forensic Science International*, 280:213–223.

Chen, Z., Song, J., Chu, W., Tong, M., and Zhao, X. (2018). A Normalized Congruent Matching Area Method for the Correlation of Breech Face Impression Images. *Journal of Research of the National Institute of Standards and Technology*.

Commons, W. (2014). Circular hough transform of four points on a circle.

Council, N. R. (2009). *Strengthening Forensic Science in the United States: A Path Forward*. The National Academies Press, Washington, DC.

Doyle, J. S. (2019). Cartridge case identification.

Fadul, T., Hernandez, G., Stoiloff, S., and Sneh, G. (2011). An Empirical Study to Improve the Scientific Foundation of Forensic Firearm and Tool Mark Identification Utilizing 10 Consecutively Manufactured Slides.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Fung, W. K. (1995). Diagnostics in linear discriminant analysis. *Journal of the American Statistical Association*, 90(431):952–956.

Gallostra, J. (2017). Augmented reality with python and opencv (part 1).

Greene, C. (2020). filt2 2d geospatial data filter. MATLAB Central File Exchange.

Heard, B. J. (2008). *Handbook of firearms and ballistics : examining and interpreting forensic evidence / Brian J. Heard.* Developments in fullerene science. Wiley-Blackwell, Oxford ; Hoboken, NJ, 2nd ed.. edition.

Hofmann, H., Vanderplas, S., Krishnan, G., and Hare, E. (2019). *x3ptools: Tools for Working with 3D Surface Measurements.* R package version 0.0.2.9000.

Hough, P. (1962). Method and means for recognizing complex patterns.

Muralikrishnan, B. and Raja, J. (2008). *Computational Surface and Roundness Metrology.* Springer Publishing Company, Incorporated, 1st edition.

Ott, D., Thompson, R., and Song, J. (2017). Applying 3D measurements and computer matching algorithms to two firearm examination proficiency tests. *Forensic Science International*, 271:98–106.

Song, J. (2013). Proposed "NIST Ballistics Identification System (NBIS)" Based on 3D Topography Measurements on Correlation Cells. *American Firearm and Tool Mark Examiners Journal*, 45(2):11.

Stockman, G. and Shapiro, L. G. (2001). *Computer Vision.* Prentice Hall PTR, USA, 1st edition.

Tai, X. H. (2019). *Matching Problems in Forensics.* Ph.d, Carnegie Mellon University.

Tong, M., Pan, Y., Li, Z., and Lin, W. (2018). Valid data based normalized cross-correlation (VDNCC) for topography identification. *Neurocomputing*, 308:184–193.

Tong, M., Song, J., and Chu, W. (2015). An Improved Algorithm of Congruent Matching Cells (CMC) Method for Firearm Evidence Identifications. *Journal of Research of the National Institute of Standards and Technology*, 120:102.

Tong, M., Song, J., Chu, W., and Thompson, R. M. (2014). Fired Cartridge Case Identification Using Optical Images and the Congruent Matching Cells (CMC) Method. *Journal of Research of the National Institute of Standards and Technology*, 119:575.

Wang, R. (2016). Image scaling and rotation: Reduction.

Zheng, X. A., Soons, J. A., and Thompson, R. M. (2016). Nist ballistics toolmark research database.

APPENDIX A. ADDITIONAL MATERIAL

This section contains the CMC distributions using the "improved" method under various pre and post-processing conditions similar to Figure 5.8, but for 7 other pre-processing conditions. Additionally, the ROC curves associated with each of these conditions are given. While nowhere near exhaustive, these do give an indication of how various pre-processing conditions interact with post-processing conditions and which combinations yield promising classification results.

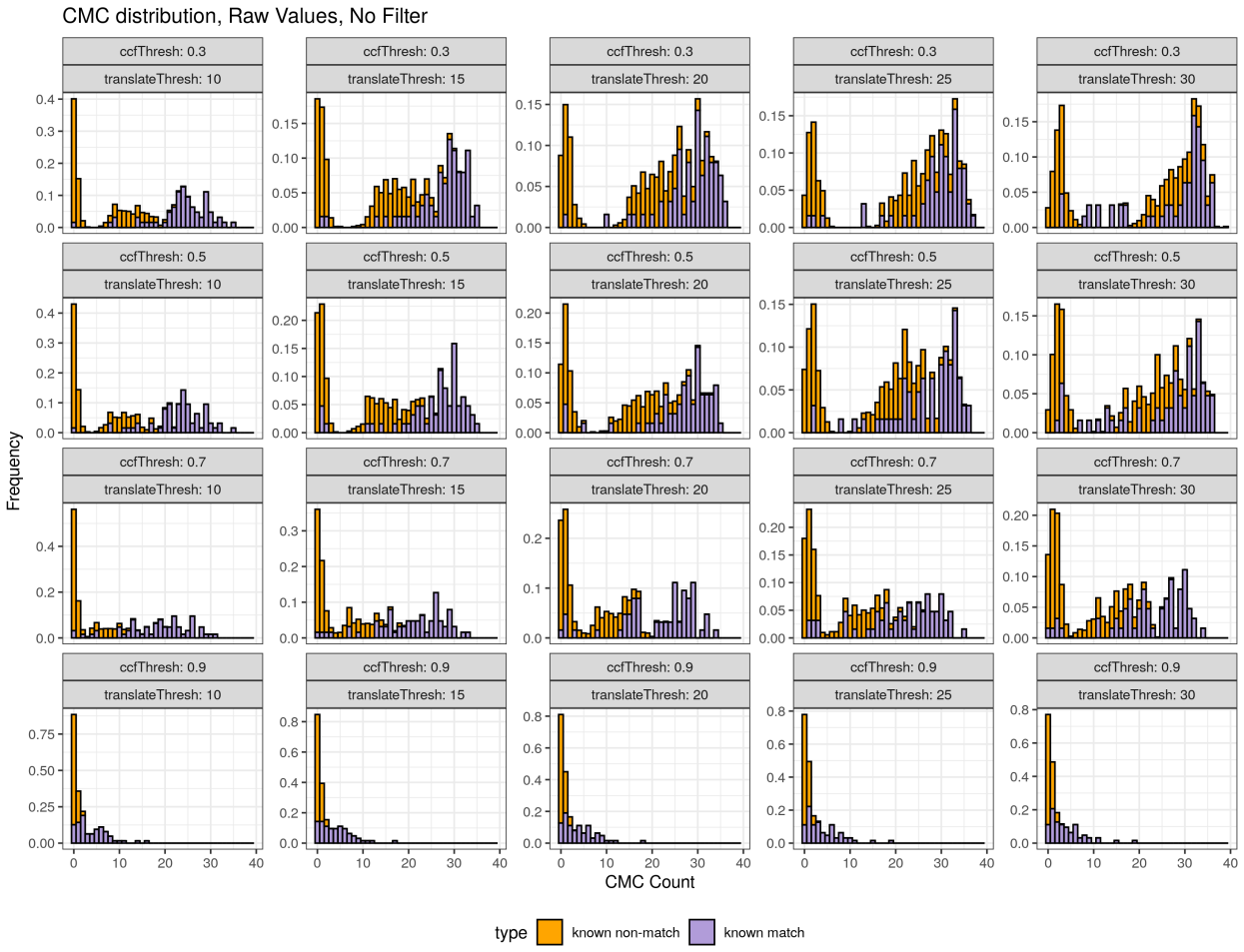


Figure A.1: CMC distributions for various filtering thresholds on non-filtered, "raw" value known match and known non-match pairs.

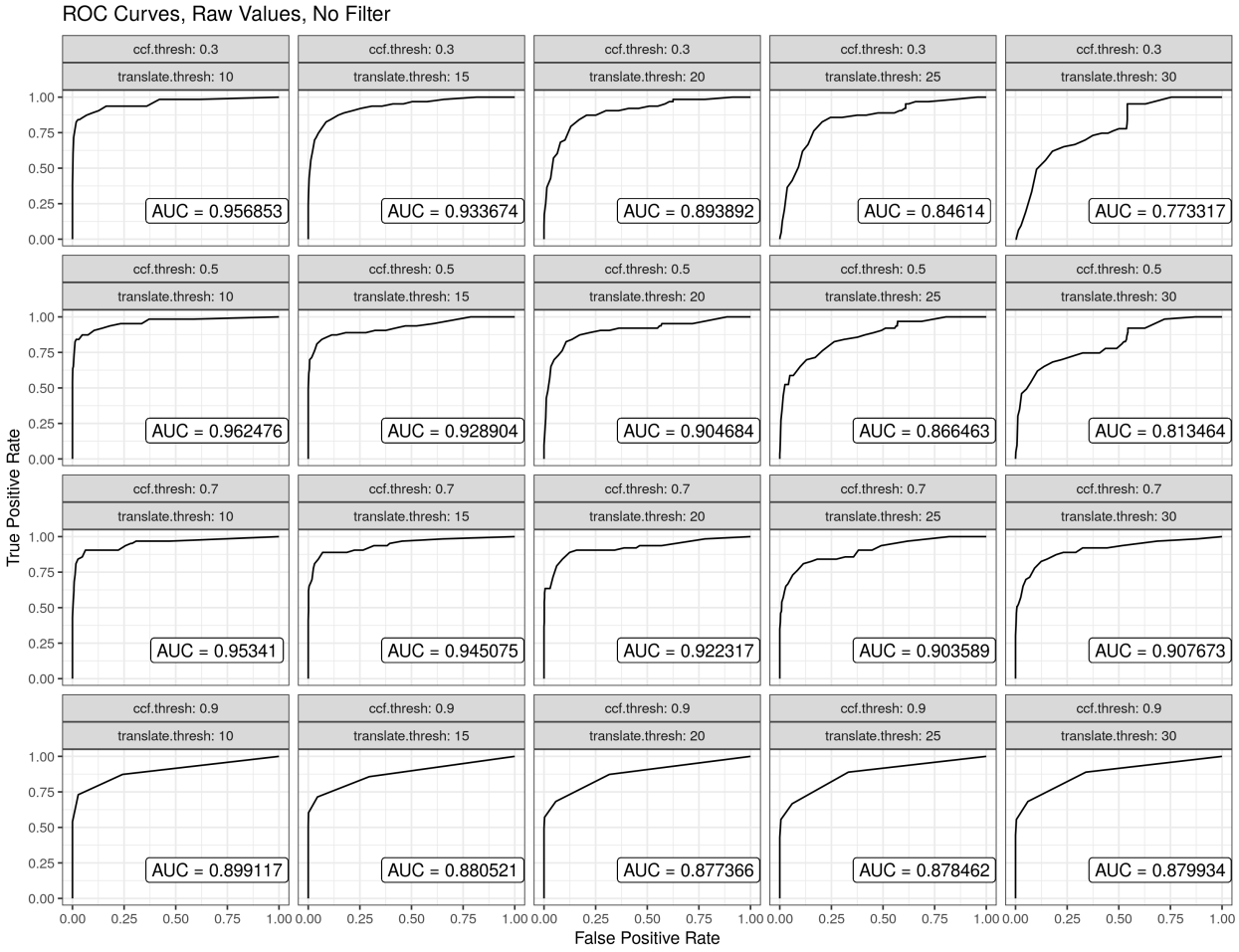


Figure A.2: ROC curves associated with the CMC distributions of Figure A.1 by varying the CMC count classification threshold

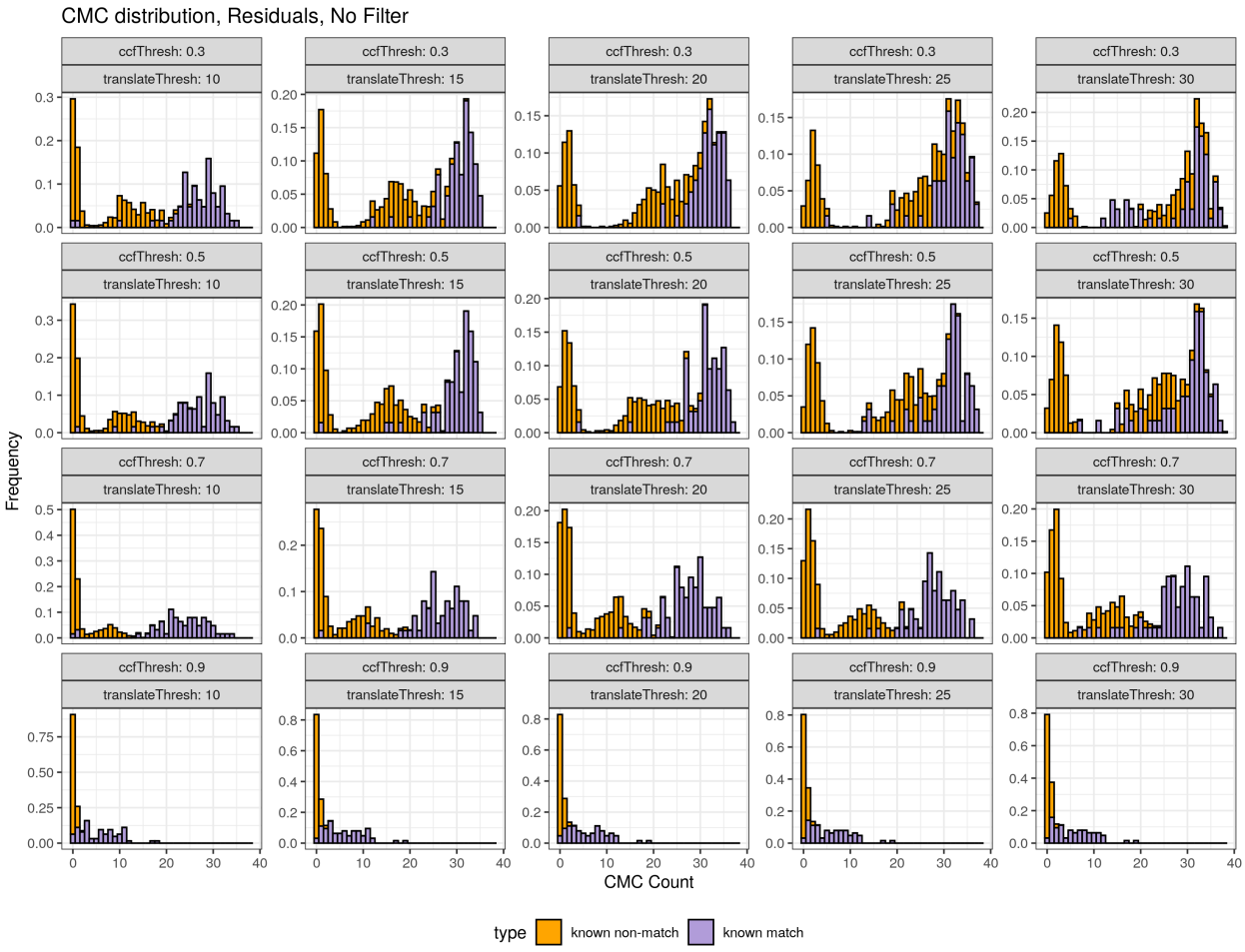


Figure A.3: CMC distributions for various filtering thresholds on non-filtered, residual value known match and known non-match pairs.

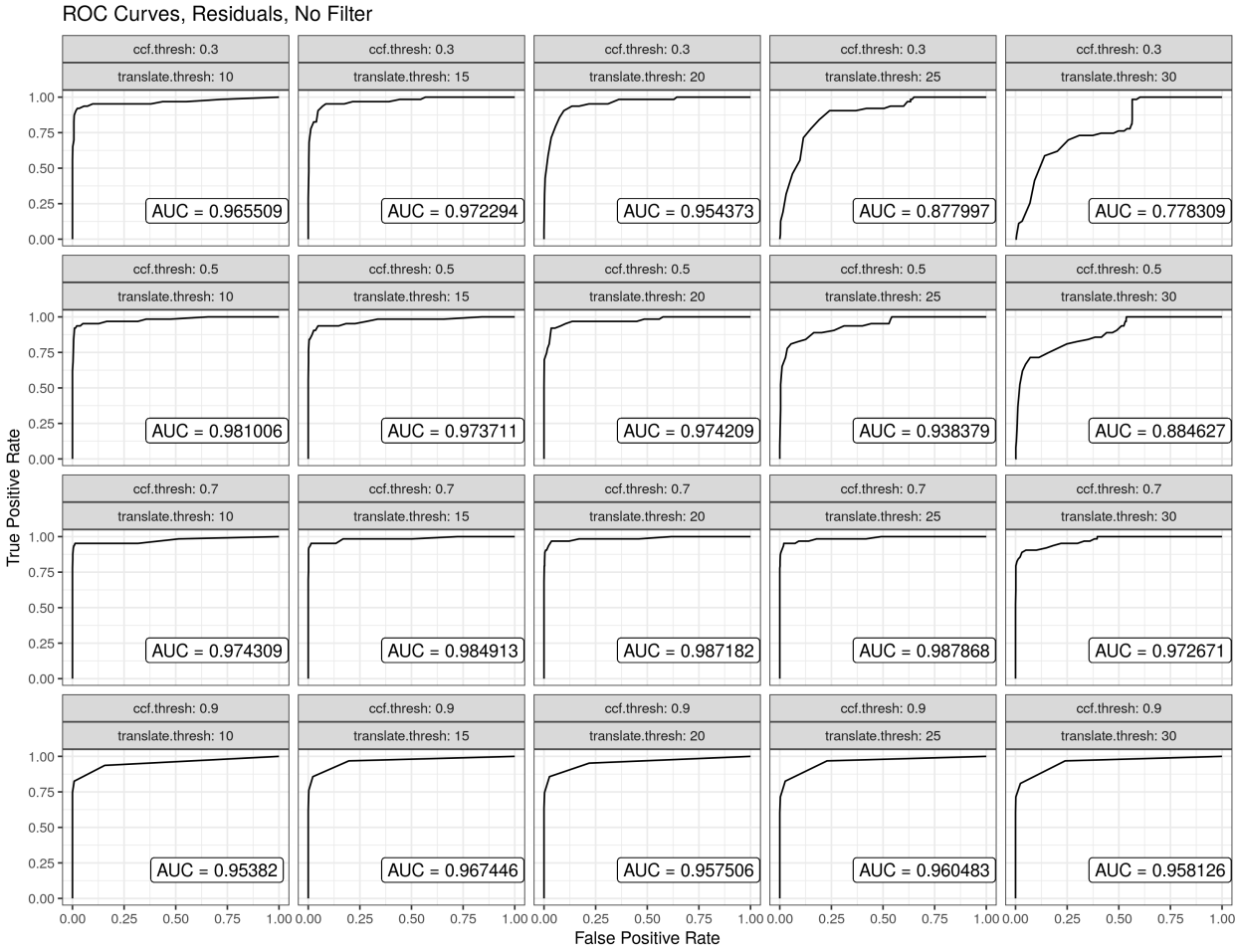


Figure A.4: ROC curves associated with the CMC distributions of Figure A.3 by varying the CMC count classification threshold

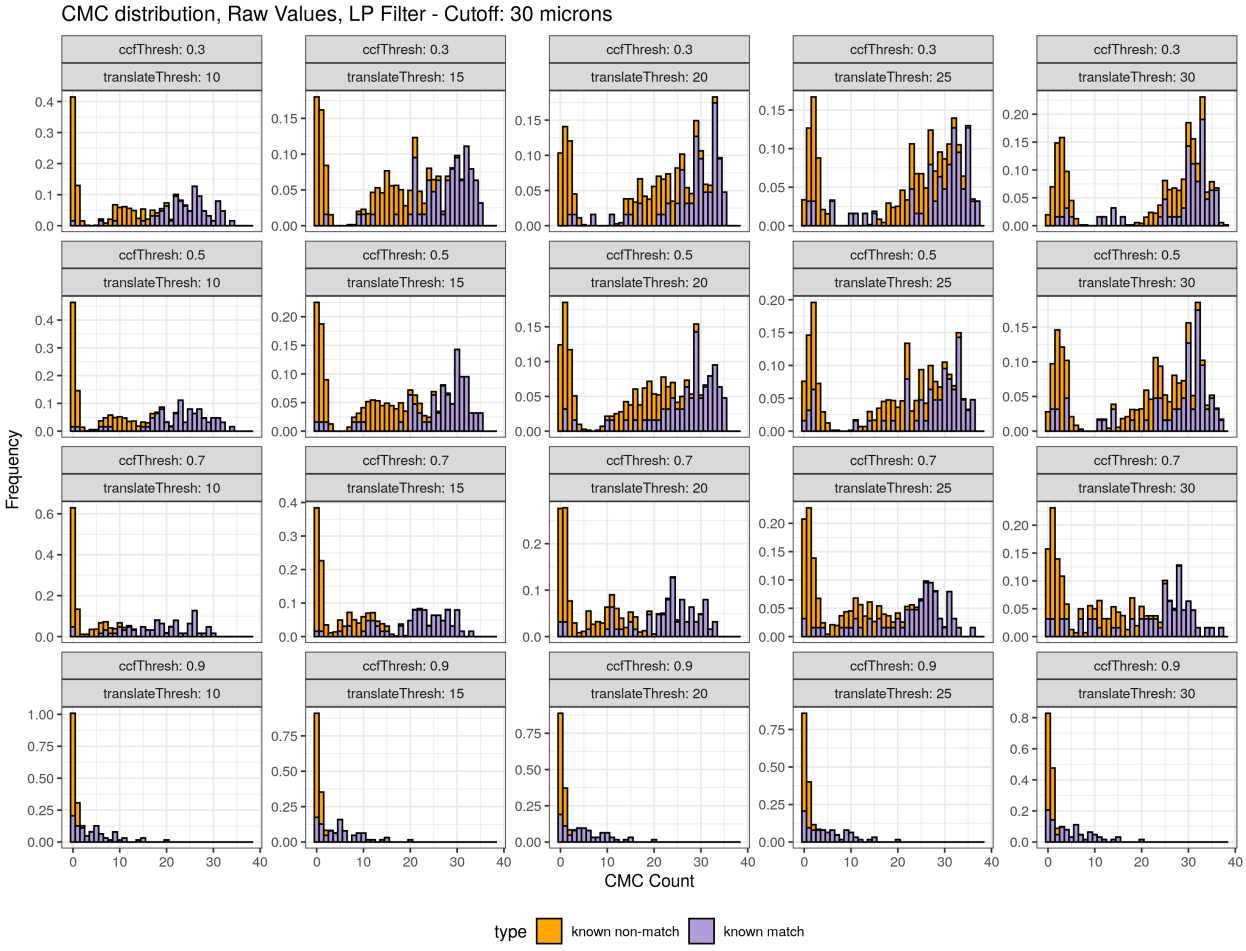


Figure A.5: CMC distributions for various filtering thresholds on lowpass filtered, "raw" value known match and known non-match pairs.

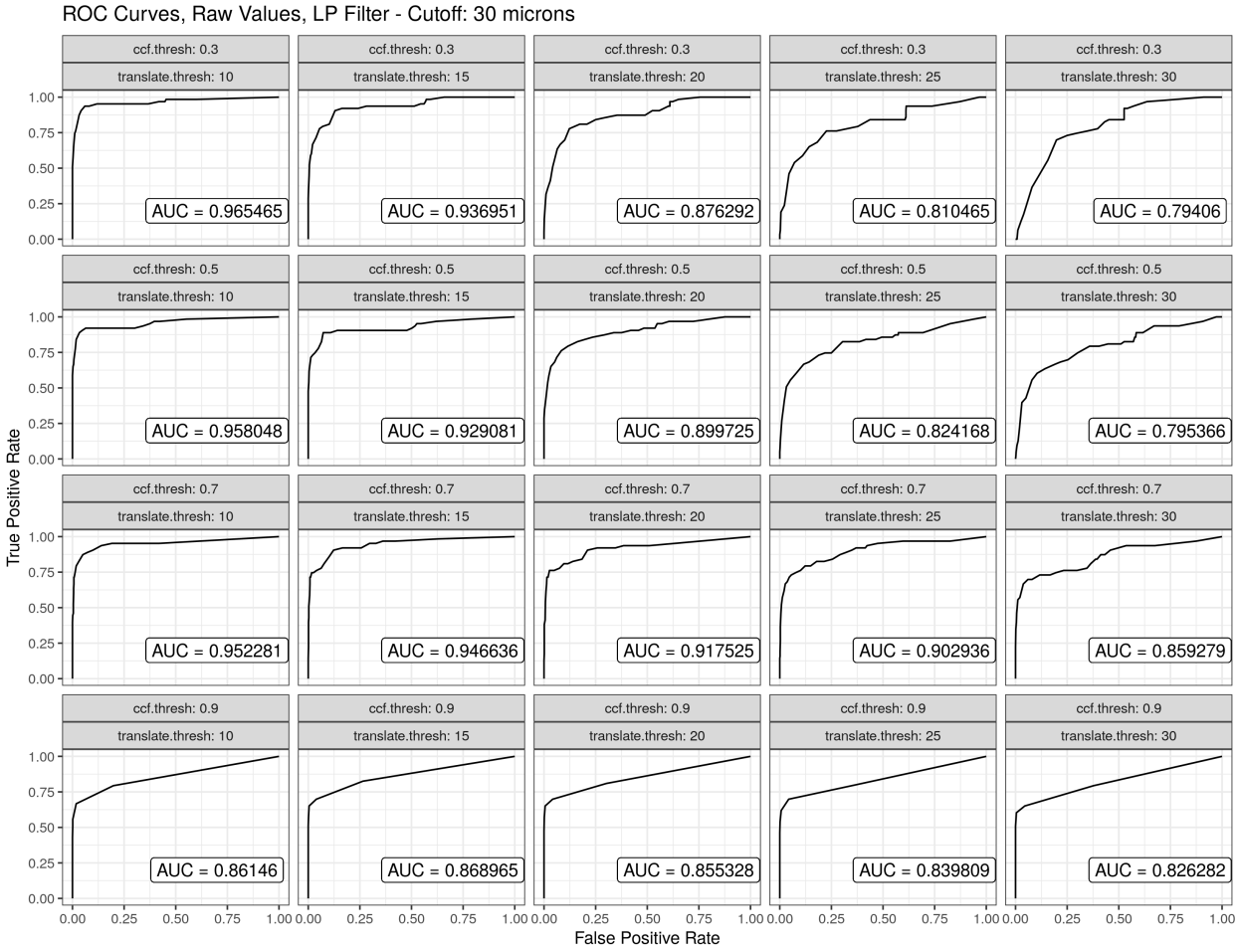


Figure A.6: ROC curves associated with the CMC distributions of Figure A.5 by varying the CMC count classification threshold

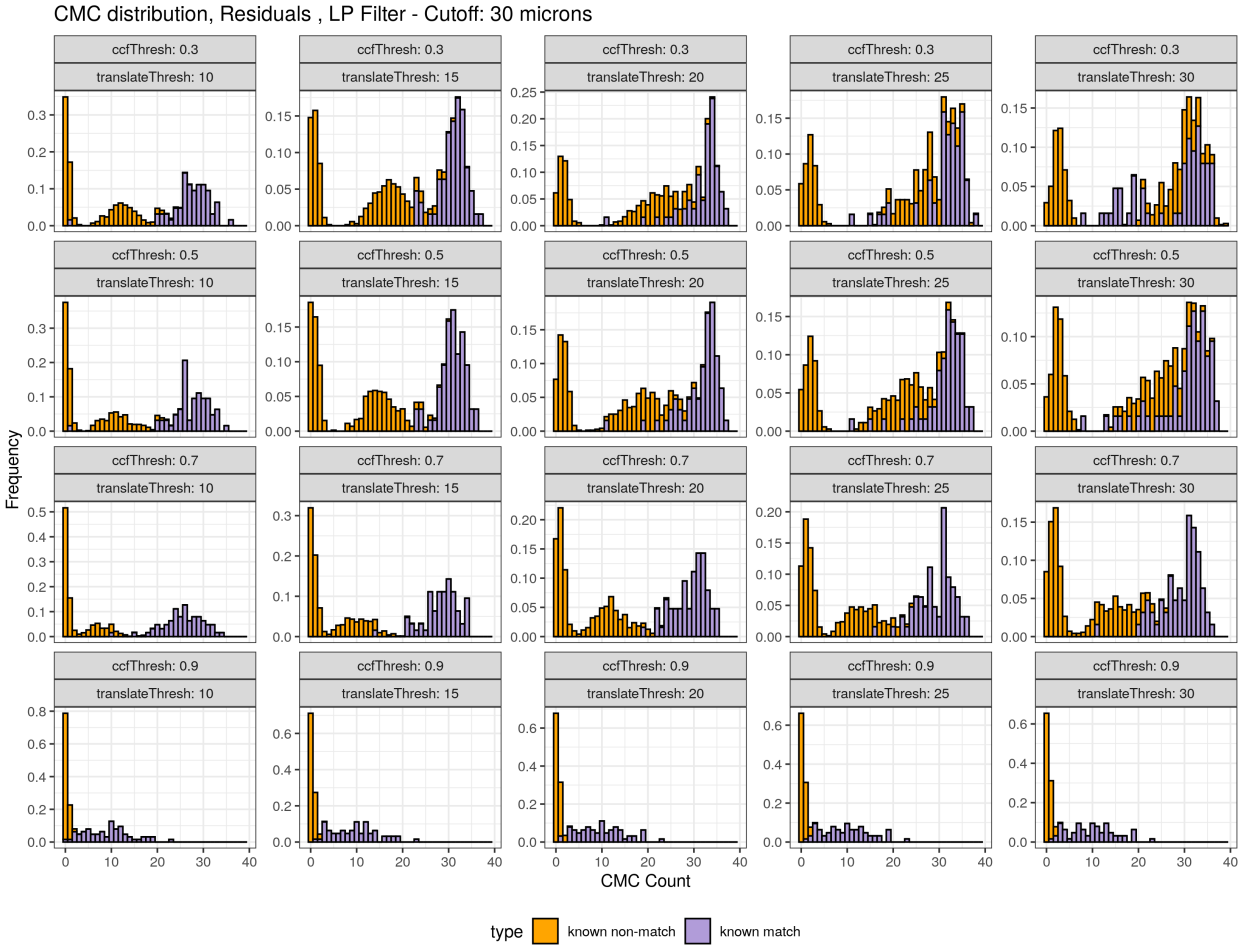


Figure A.7: CMC distributions for various filtering thresholds on lowpass filtered, residual value known match and known non-match pairs.

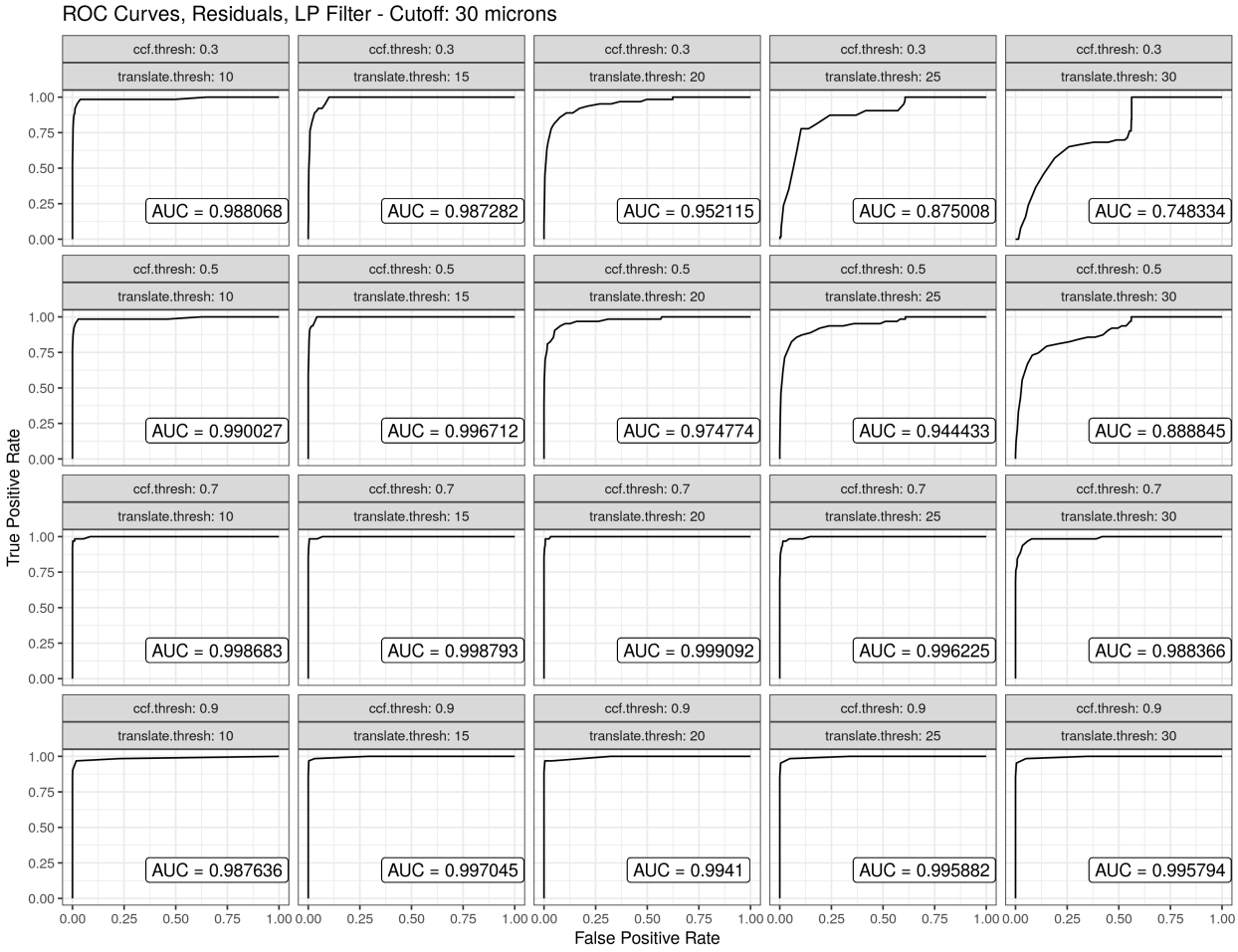


Figure A.8: ROC curves associated with the CMC distributions of Figure A.7 by varying the CMC count classification threshold

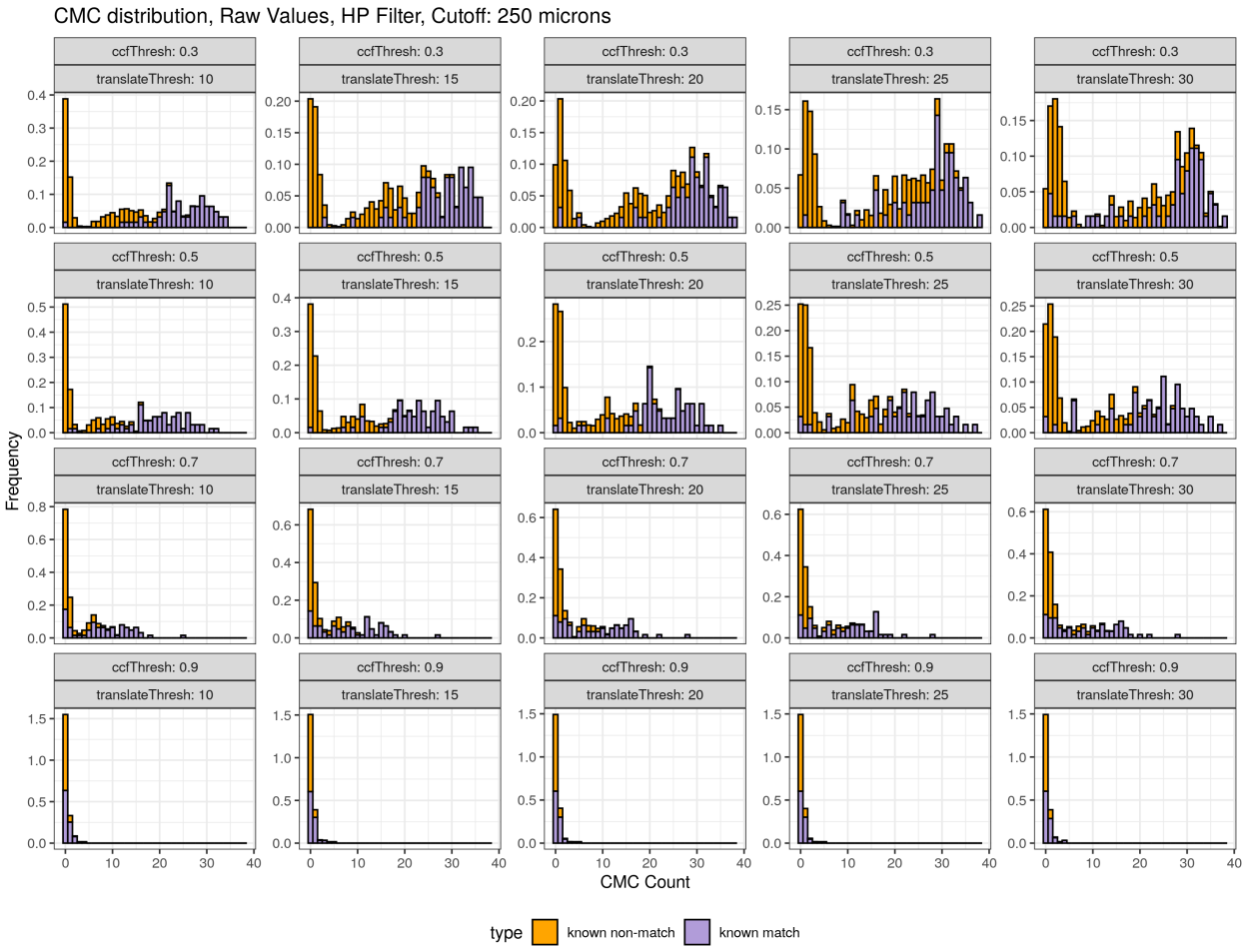


Figure A.9: CMC distributions for various filtering thresholds on highpass filtered, "raw" value known match and known non-match pairs.

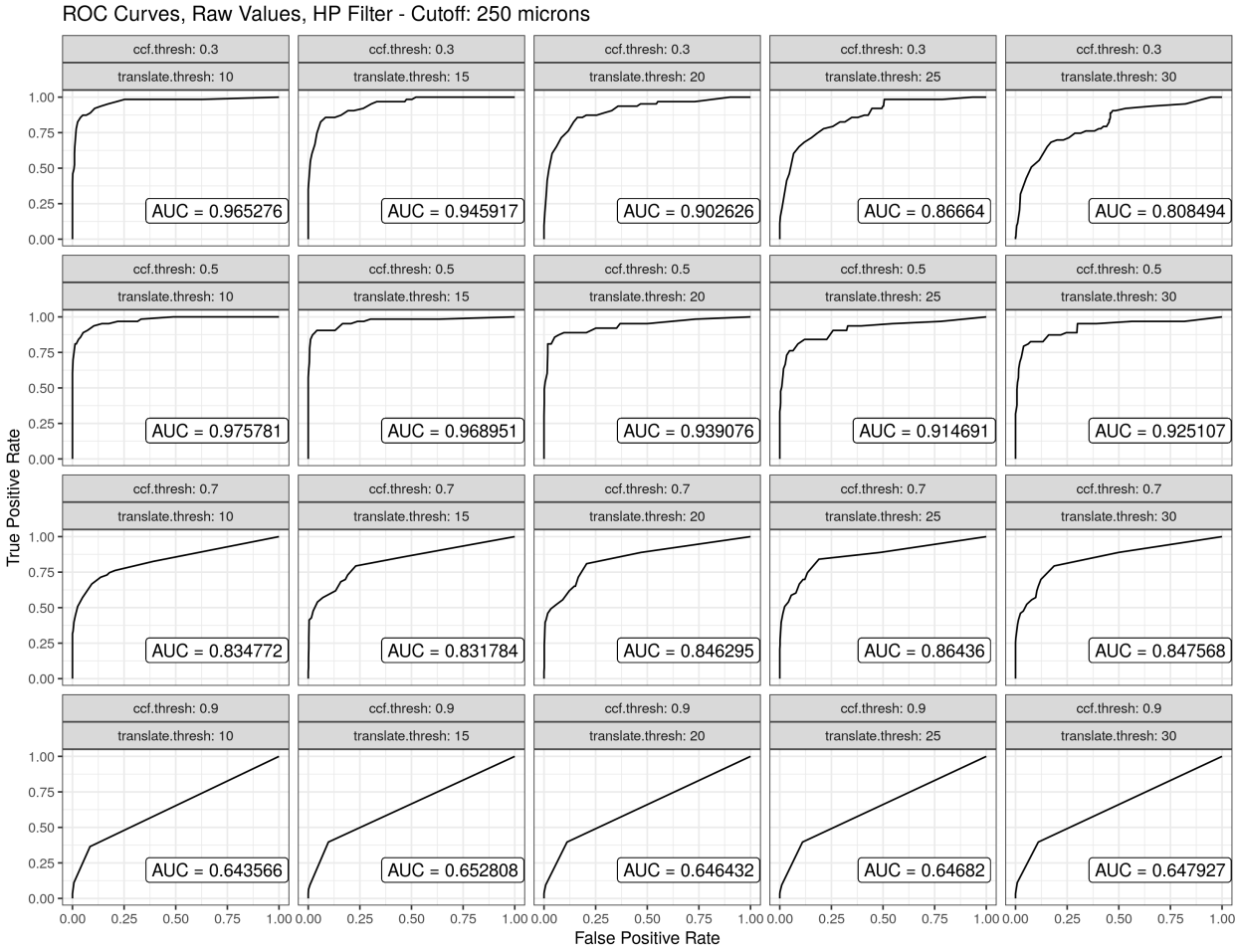


Figure A.10: ROC curves associated with the CMC distributions of Figure A.9 by varying the CMC count classification threshold

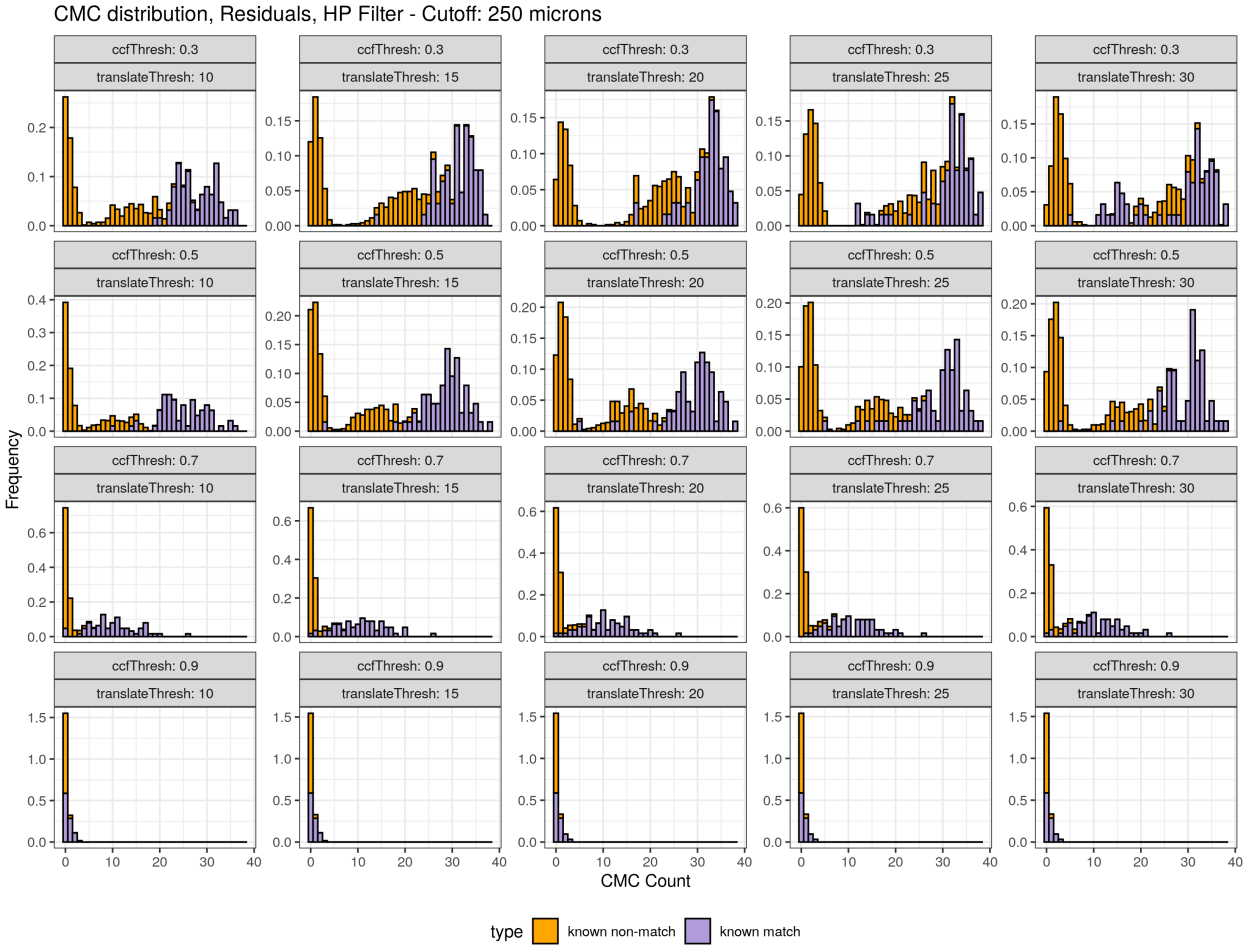


Figure A.11: CMC distributions for various filtering thresholds on highpass filtered, residual value known match and known non-match pairs.

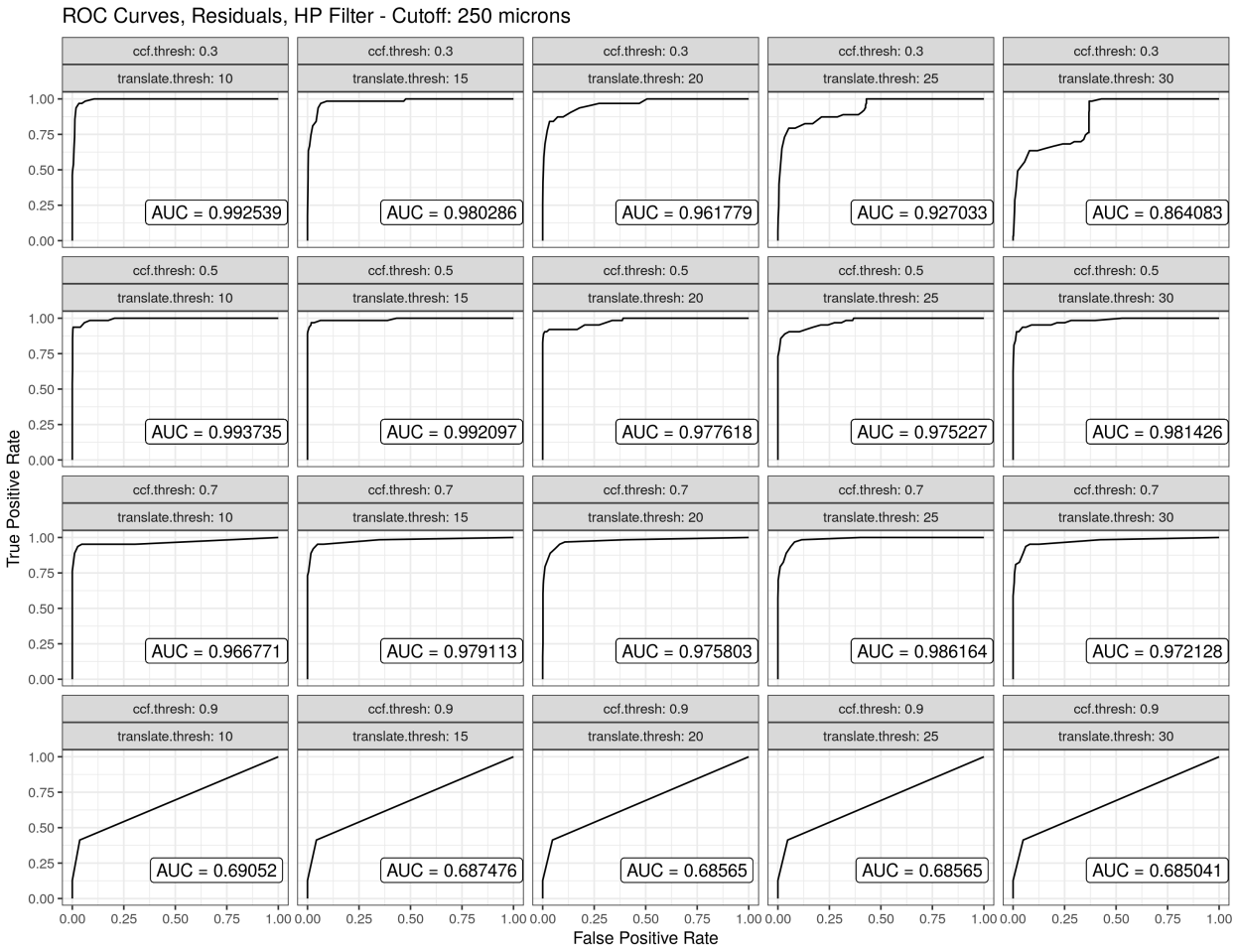


Figure A.12: ROC curves associated with the CMC distributions of Figure A.11 by varying the CMC count classification threshold

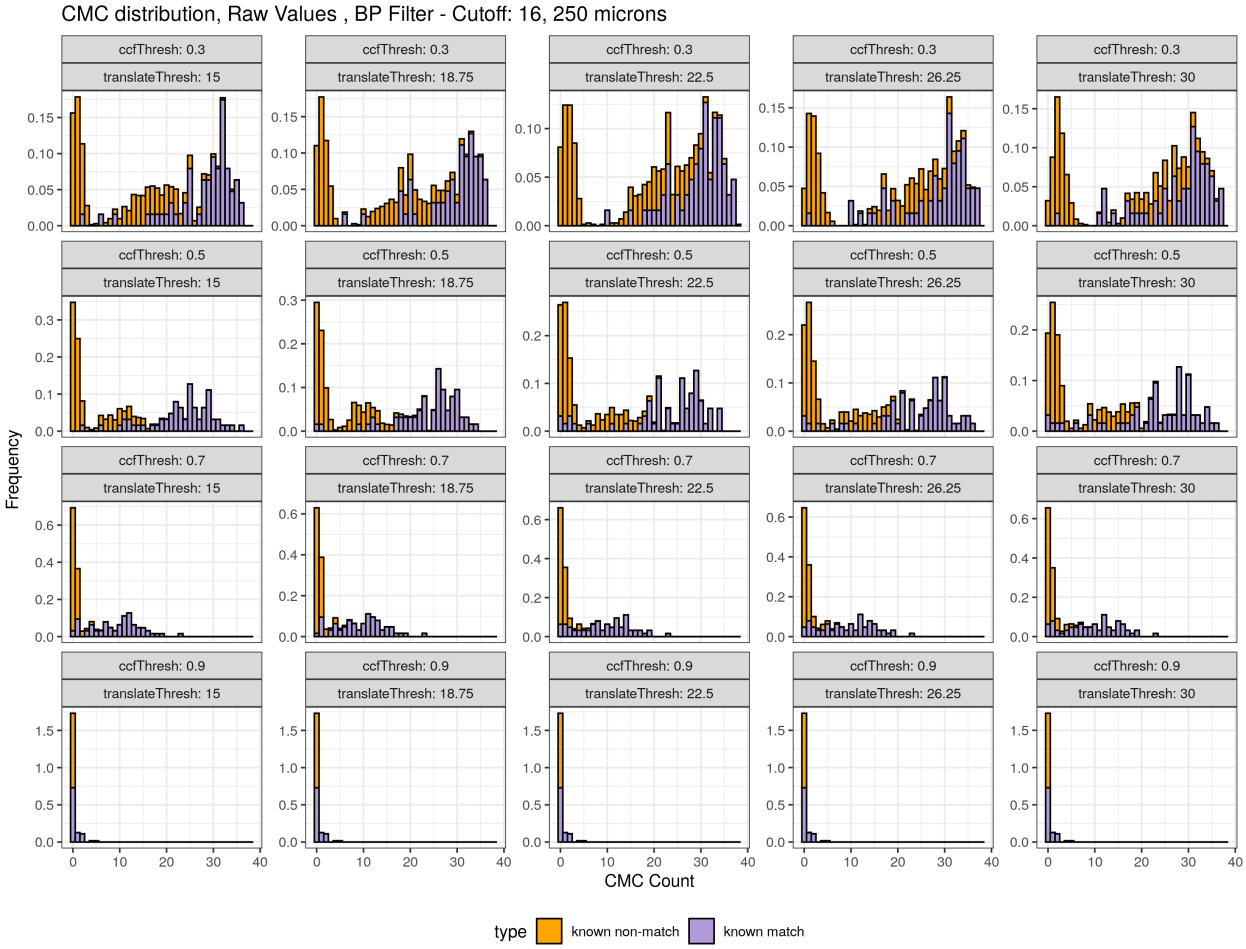


Figure A.13: CMC distributions for various filtering thresholds on bandpass filtered, "raw" value known match and known non-match pairs.

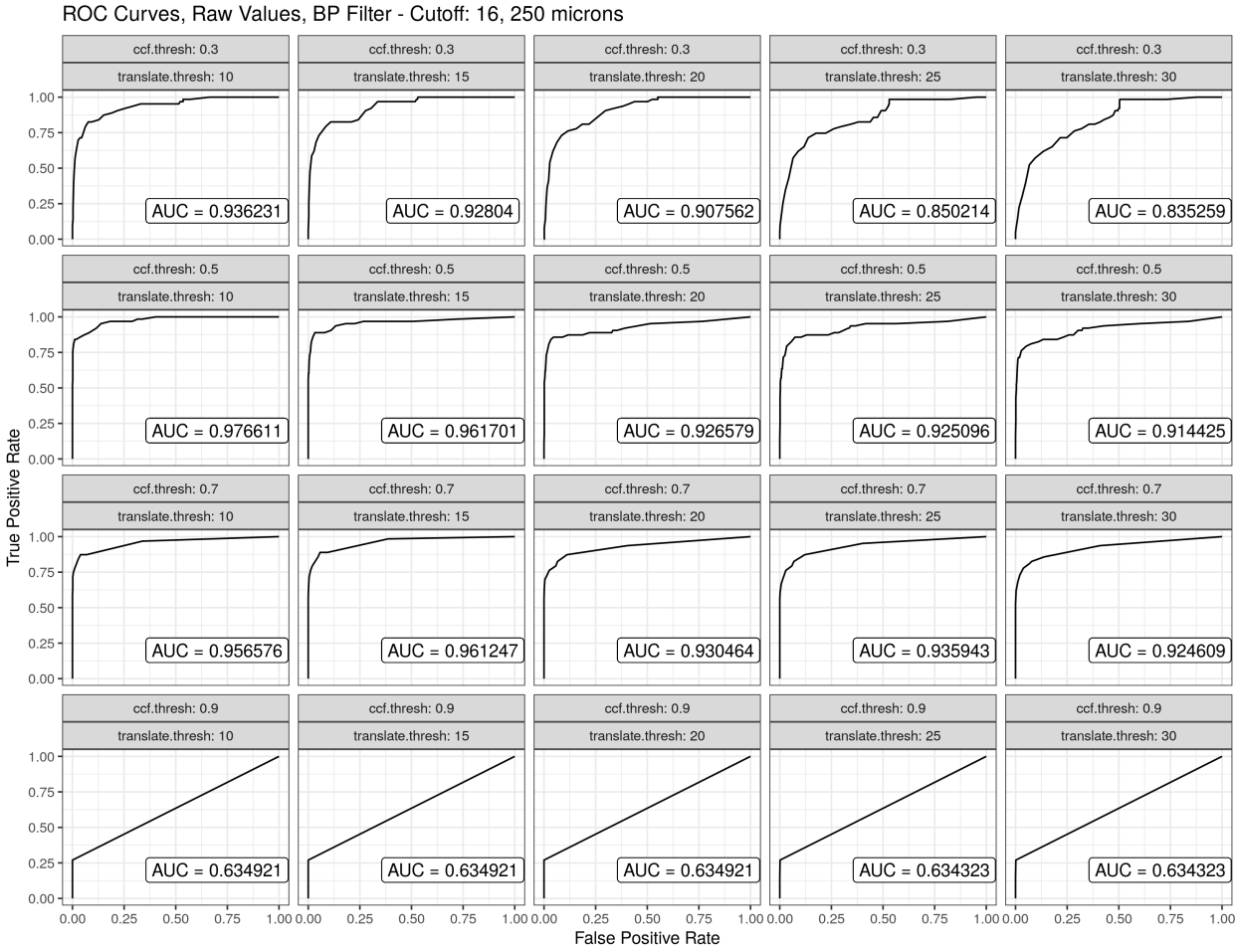


Figure A.14: ROC curves associated with the CMC distributions of Figure A.13 by varying the CMC count classification threshold