Univeristy of Tartu

Faculty of Science and Technology

Institute of Technology

Rain Eric Haamer

Automated Data Extraction and Analysis for Arrayed Primer Extension Images

Bachelor's thesis (12 ECTP)
Computer Engineering Curriculum

Supervisor:

Assoc. Prof. Gholamreza Anbarjafari

Tartu 2017

Automated Data Extraction and Analysis for Arrayed Primer Extension Images

Abstract

The Arrayed Primer Extension (APEX) method is used to detect Single-Nucleotide Polymorphism (SNP), deletion and insertion based diseases. The main method consists of washing different flourophores over oligonukleotites and then analysing which flourophore attached to identify the oligonukleotites. A crucial step in the APEX method is analysing the captured light from the flourophores when a laser excites them. The current method of analysing this data requires extensive manual review.

This thesis describes a method of automating the data grid detection in the captured images which is currently a manual task. The second part describes the application of predictive methods for the data analysis of the four captured images and comparisons to the already existing clustering method.

CERCS: T111 Imaging, image processing

Keywords: KLD, SVM, LBP, Clustering

Praimerekstensioon Oligonukleotiidmaatriksi Piltidel Olevate Andmete Tuvastuse ja Analüüsi Automatiseerimine

Lühikokkuvõte

Praimerekstensioon oligonukleotiidmaatriks (APEX) meetodit kasutatakse ühenukleotiidilisi polümorfismil (SNP), deletsioonil ja insetsioonil põhinevate haiguste tuvastamiseks. APEX meetodi põhiidee seisneb oligonukleotiidide pesemises erinevate fluorestseerivate markeritega ning oligonukleotiidide määramiseks analüüsitakse markerite kinnitumist. Kinnitunud markerite tuvastuseks kasutatakse fluorestsentse ergutavad lasereid. Kriitiline etapp APEX meetodis on laseri abil tehtud piltide analüüs, mis hetkel tehakse manuaalselt.

See tees koosneb kahest peamisest osast. Esimeses osas on välja toodud meetod, mis automatiseerib pildil olevate andmete ruudustiku asukoha määramist. Teises osas analüüsitakse erinevaid SVM baasil ennustavaid mudeleid, mis suudavad vastuseid paremini määrata, kui hetkel kasutuses olev meetod.

CERCS: T111 Pilditehnika

Märksõnad: KLD, SVM, LBP, Klusterdamine

Contents

List of Figures

List of Tables

Acronyms

APEX Arrayed Primer Extension. 2, 6, 7

CHT Circle Hough Transform. 15, 23

FCM Fuzzy C-means. 19–21, 24

HDR High-Dynamic-Range. 6, 7

KLD Kullback–Leibler Divergence. 15, 23

LBP Local Binary Pattern. 13, 17, 18, 21, 25–27

PCR Polymerase Chain Reaction. 7, 28

RI-LBP Rotation-Invariant Local Binary Pattern. 17, 25, 26

SNP Single-Nucleotide Polymorphism. 2, 6

SVM Support Vector Machine. 7, 17–21, 24, 25, 28

1 Introduction

The APEX method is used to detect different types of genetic mutations and polymorphisms caused by SNP, deletion and insertion [27]. A detailed description of the APEX method itself can be found in section 1.1. One step in the APEX process generates grayscale High-Dynamic-Range (HDR) images that contain a small dot grid, which have to be analysed by trained professionals. Unfortunately this is very time consuming as the grids have to first be assigned and adjusted manually. Because the current programs meant to aid in the labeling process are not very accurate, the tests have to also be manually reviewed.

The biggest problem with the generated images for automated systems is how hard it is to actually detect and align the data so each cell in the grid can be analysed separately. The images vary both in resolution and luminosity, and the background noise can be so overpowering that even aligning the grid manually can be a difficult task. The grids do have specific corner markers meant to aid in the detection, but in almost half of the cases, some of the markers are either indistinguishable from noise or just missing entirely.

There are other methods out there that do both dot pattern [8, 13, 34] and circular dot detection [6], but those methods either require the grid to contain most of its dots or all of the aiding markers. Sadly neither of these criteria can't be guaranteed in this case, as part from often missing the corner markers, the images rarely have more than 40% of the dots detectable on a single image.

Leaving the dot grid detection aside, the analysis of the data may seem like a simpler task, but this is not the case. Because of the background noise being unique for each image, the overall values of each dot varies quite significantly. Coupled with the fact that some dots in particular act differently from others, with much higher or lower thresholds than normal, makes the manual assignment of thresholds a fruitless task.

Traditionally the most successful methods applied in similar cases have been manually adjusted clustering algorithms. [19,29] Clustering has provided reasonable results for tools like the MACGT which could provide accuracy of around 98% and even up to 99.5% with adjusted confidence measures [33]. There also exist some methods that employ supervised learning based methods like GetGenos [14] which review the quality of primer extension 2-color fluorescent reactions. However, the methods mentioned are not plagued by large amounts of noise, so the quality measures there can focus more on the circularity of the dots and just discard those that are not.

Because the main problem in this case is the classification of dots in a noisy image and not the actual data behind it, similarities can even be drawn with methods which try to gauge the quality of ball grid array solder joints on printed circuit boards [17].

This thesis presents two main parts, the first one employs a method which automatically locates and calculates the translation, rotation and scaling of a fixed size dot grid array in the image with error corrections. This allows for the manual process of grid alignment to be completely automated even in images with high noise.

The second part tackles the problem of predicting the dot values through the use of clustering

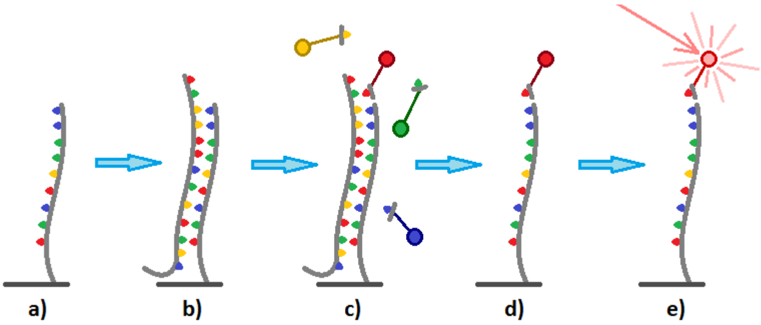

Figure 1.1: Illustration of the APEX reaction where DNA fragments are annealed to a previously designed chip(ab). A solution of terminator nucleotides with flourecent markers is left to bind with the oligonukleotites(c). After the reaction the rest of the markers and DNA fragments are washed off(d) and a laser is used to excite the flourophores(e).

and trained Support Vector Machine (SVM)'s. Several variations in input data are tested and compared against each other, with the best being compared against the existing method.

1.1 APEX method brief

Arrayed Primer Extension (APEX) is a method in which complimentary oligonukleotites are immobilised on a test glass surface. The test sample DNA is then amplified using Polymerase Chain Reaction (PCR), digested enzymatically and hybridisized with the oligonukleotites. A solution of four dye-labeled fluorescent marked terminators (ddATP, ddCTP, ddGTP and ddUTP [25]) are left to bind with the complementary ends of the oligonukleotites. [18, 21, 30] Four different dye terminators are used which allows for simultaneous evaluation of possible nucleotide changes. [27] The terminator solution is then washed off and the oligonukleotites are exited using 4 lasers reflecting off the test glass. [32]

The 4 lasers used in Genorama® QuattroImage^{TM} [20] are $635\ nm$ Diode, $594\ nm$ DPSS, $532\ nm$ DPSS and $473\ nm$ DPSS, corresponding to the four fluorescent markers.

The exited flourophores are recorded in four images corresponding to the four lasers using a cooled digital CCD camera system with a resolution of 2184×1472 with each pixel representing a length of $6.8\ \mu m$. The four captured images are then analysed and compared to previous results in order to reduce interference from defective grid locations. The results of each test are then matched with a database with the corresponding translations for each of the mutations.

1.2 Current image analysis

The method currently in use for analyzing the four test images consist of loading the data into Genorama$^{\text{TM}}$ 4.5 genotyping software [20] which illuminates and displays the HDR image data. The grid is then placed on the images manually by marking each of the four corners and then an automatic analysis predicts most of the outputs for each of the data points using a customised clustering algorithm. [31]

The analysis program does not consider the previous test results of the corresponding matrix positions but only the unorganized values of the current test. Because of this, the test results lose accuracy when dealing with abnormal data points which routinely should have higher or lower threshold values to be considered positive or negative.

Each of the test results have to be reanalyzed by a trained professional in order to eliminate mislabeled results, because the current automated method is not accurate enough for full automation.

2 Acquired data

All of the image and label data was acquired from Asper Biotech for research purposes only. The files corresponding to 500 tests were generated over a 4 year time-span starting from 2012. The data is unordered outside of the year it was made in and does not contain any information that could tie them to the original patients. During the initial data processing, the names of the tests were set as unique numeric variables, because of similarly named files.

2.1 Image data description

The image data is in the form of raw 16 bit uncompressed monochrome images that have an average size of 640×817 with $\sigma_x = 70.1$ and $\sigma_y = 59.5$. The bit depth makes analysing the images by hand very difficult without the use of proper illumination enhancement methods. Each of the images contain a 24×16 dot grid of data with an unknown location, size and rotation. Generally the size and rotation of the grids varies very little. The datapoints in the grid come with corresponding duplicate pair dots next to the original on the X axis. The duplicates are there to counteract the effect of noise on the final evaluation. The grid itself comes with 4 corner markers that are always set to give a positive output along with their own pair dots. For each test the final 36 datapoint locations are empty, with the corner markers taking up 4 datapoint locations each. This means each test contains 664 evaluatable data points where each of the points can only occupy one binary state.

Due to human error or heavy noise, the corner dots are not always apparent and the bottom corner markers are often completely missing from the dataset.

The borders of the data points rarely have sharp transitions due to background noise. They also have varying diameters of about $6 - 20$ pixels depending on how the edge is defined as the illumination of the dots is inconsistent. The distance between each dot is $19 - 25$ pixels (The

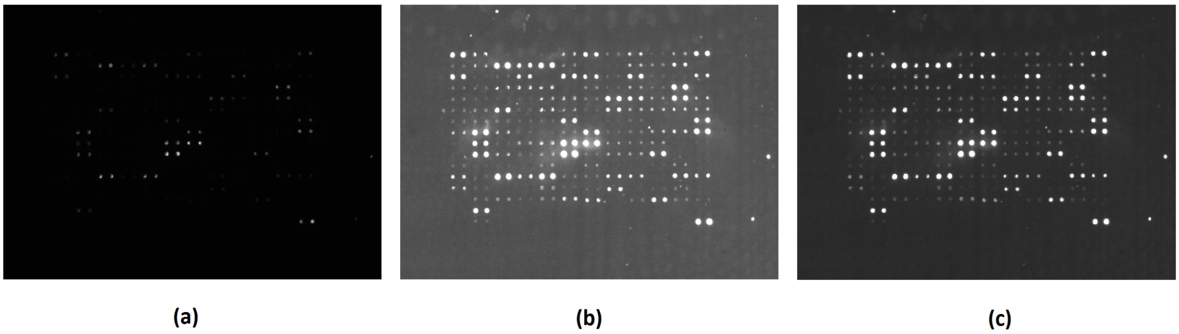

Figure 2.1: (a) is a dataset image converted to have a depth of 8 bits. (b) has had its illumination enhanced using Singular Value Equalization and (c) is enhanced using Discrete Wavelet Transform and Singular Value Decomposition [10, 11].

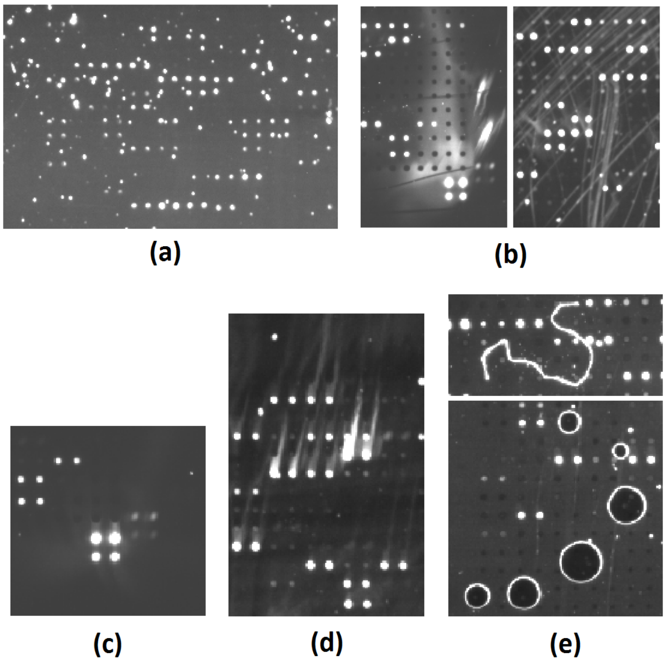

Figure 2.2: Examples of noise found in the data images. (a) contains an entire dataset completely hidden behind heavy amounts of pepper noise. The 2 images in (b) show the 2 most common types of background anomalies. (c) and (d) are very common examples of shadow and bleed noise respectively. The final 2 images in (e) show larger anomalies which are less common than other types of noise.

distance only varies between different tests) and the rotation of the grids doesn't exceed ±10°.

2.1.1 Noise

Each of the images contain a substantial amount of noise which can be categorized into 5 separate groups:

- **Pepper noise**, if the cleaning solution has been contaminated or the wrong one is used. This type of noise is very common and can have a devastating impact on the dot grid as seen in Fig. 2.2a.
- **Background anomalies**, where the entire background of the test glasses have either a gradient or underlying texture.
- **Shadow noise** in which duplicates of very vibrant data points carry over to neighboring matrix positions when the cleaning solution has been over-saturated.
- **Bleed noise**, where very vibrant data points bleed their values over to neighboring cells.
- **Larger anomalies** like small dust particles, hair follicles or unknown streaks in the dataset. This type of noise is very uncommon relative to the other types but like in Fig. 2.2e it can make several positions in the dataset completely unreadable.

2.2 Evaluation data description

All of the acquired test images also have their corresponding results which have been manually reviewed by a trained professional. The result data is separated into 4 files, separated by year, with the tests from 2012 been distributed into the data files of 2011 and 2013.

The columns in each of the files represent a single test and each of the rows represent the index of a cell in the grid starting from 1. Each cell in the results table contains a string with a separating forward slash which denotes whether it is the odd or even Y index respectively.

The results themselves are as different combinations of the characters A, T, G or C including none which is represented as a ”$-$”. In some cases the output cell contains an exclamation mark. The exclamation is a leftover from the original software predictions as it denotes locations in the data where the result is different from the expected, notating a mutation. As some of the exclamation marks have been deleted in the manual evaluation process, these can be discarded as it lacks any effect on the result.

The order of the characters in each of the cells does not have an effect on the meaning but the preferred order for the sake of consistency is $[A, C, G, T]$.

3 Data extraction

3.1 Image cleanup

For all of the following functions, the base image will be annotated as I^b.

In order to remove high frequency noise and reduce the amount of small pepper noise, a small Pillbox filter is applied over the image. By applying a threshold with the 95th percentile, a binary mask called I^t is created where most of the low values along with the background have been removed.

$$I^t = \begin{cases} 1, & \text{if} \quad i^b > I^b_{95th} \\ 0, & \text{Otherwise} \end{cases} \quad i^b \in I^b \tag{3.1}$$

Using Matlabs built in *regionprops* function, the centers of white circular blobs in I^t are located. The array of those center points will be denoted as D.

For each of the found points, a vector is calculated from one point to another.

$$\overrightarrow{v_{ij}} = d_i - d_j \quad \{d_i, d_j\} \in D \tag{3.2}$$

In order to reduce the amount of noisy dots, all points that do not have a pair dot are discarded. In order to accomplish this, all of the vectors calculated in function 3.2 with $19 < ||\overrightarrow{v_{ij}}|| < 25$ and $-10° < \theta_{\overrightarrow{v_{ij}}} < 10°$ are added to a new vector array which will be denoted as $\vec{G}$, the rest will be discarded. The original image is then rotated by $-\theta_{\vec{g}} \quad \vec{g} = \mu_{\vec{G}}$, which will align all of the data for further processing.

Two new binary masks are created on the rotated image with the threshold values of the 95th percentile and 99th percentile with the aim of both increasing the amount of and recalculating the previous datapoints. For the two new masks, the the center points are located using the same method as before. The points that form a vector with $19 < ||\overrightarrow{v_{ij}}|| < 25$ and $-3° < \theta_{\overrightarrow{v_{ij}}} < 3°$ are then added to filtered point arrays which will be called P_{95} and P_{99} representing the data in the 95th percentile and 99th percentile threshold images respectively. The vectors themselves that fulfill those requirements will be added to $\overrightarrow{G_{95}}$ and $\vec{G}_{99}$ respectively, the union of those vectors will be annotated as $\vec{G} = \overrightarrow{G_{95}} \cup \overrightarrow{G_{99}}$. The two point arrays are then merged together into $P \subseteq P_{95} \cup P_{99}$ so that $\nexists p^a, p^b \in P$, $\overrightarrow{p^a p^b} < 0.7 \cdot ||\overrightarrow{\mu_G}||$. In order to reduce the amount of outliers, points p^a where $\nexists p^b \in P$, $\overrightarrow{p^a p^b} < 9 \cdot ||\overrightarrow{\mu_G}||$ are removed from P.

The impact of these prunings can be seen in Fig. 3.1 where the image (a) contains all of the points in $P_{95} \cup P_{99}$ and the image (b) contains the points in the final P.

3.2 Grid alignment

After the original image has been cleaned and some point locations in the dot matrix have been located, a grid must be placed on the dot matrix in order to isolate the points into separate cells.

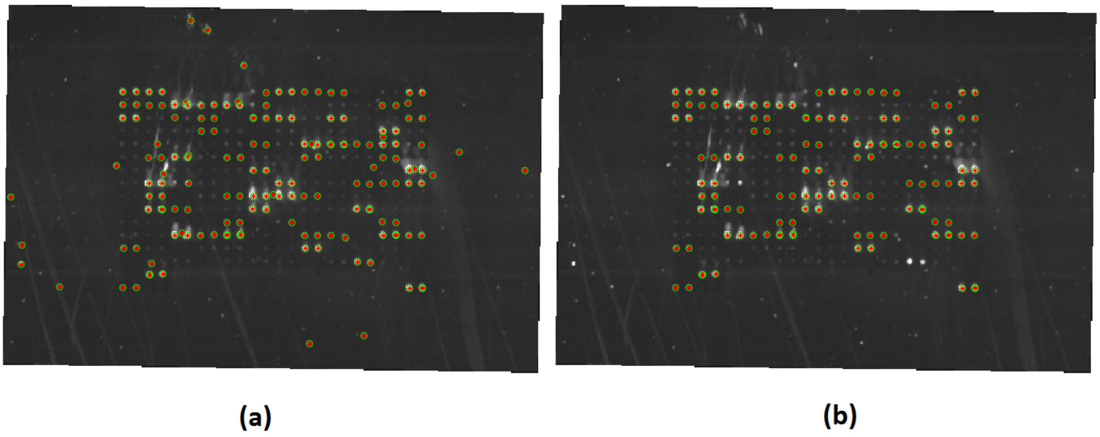

Figure 3.1: All of the raw detected centers of each blob in the image are shown in (a), while (b) depicts the same image but in this case, all detected points that don't fulfill the pair criteria or are classified as outliers have been removed. Both of the images have already been rotated using $\theta_{\vec{g}}$.

This will allow for comparisons between different point areas based on the point index when using local binary pattern (Local Binary Pattern (LBP)) or local histograms based methods [3, 12]. This will also be further needed when locating the actual data point locations for the sampling of center values. In order to place the grid on the dot matrix, 3 separate methods are proposed and tested. The first method approached the grid placement as straight forward bounding box problem while the other two apply more complex approaches.

3.2.1 Bounding boxes

In order to place the grid on the dot matrix, two bounding boxes are applied over the points of P with the first bounding box receiving the minimum and maximum X and Y values from P as the bounding area. By discarding points with the 3 highest and lowest X and Y values in P a second bounding box is also created with the intent of reducing the effect of any remaining tilt in the dot grid. The average of these 2 bounding boxes is then used to create the final bounding box for the grid. The grid itself is sectioned evenly with each cell taking the size of $\frac{P^x}{24} \times \frac{P^y}{16} \quad \{P^x, P^y\} = |\vec{h}|$ where $\vec{h}$ is the vector from one corner of the bounding box to another.

3.2.2 Image alignment

A less straight forward approach for finding the grid location in the image consisted of using a combination of linear and non-linear registration transformations, more specifically the Mathlab function called *imregtform*. The *imregtform* function compares 2 similar images and returns the translation, rotation and scaling of the second image for it to fit on the first image. The goal of this method is to to align a perfect dot grid onto the image so the transformation of the dot grid in the image could be calculated.

Firsly a black image is created and all of the pixel locations described in P are set to single white values. For the second image a black image is populated with a uniform grid of white points corresponding to the data grid size of 24×16. All of the points in the second image are also represented as single dots with the distance between each point being 21 pixels.

The images are then matched together and with the transformation parameters gained from the second images relation to the first, the dot grid location of the first image is defined. This further allowed a grid to be placed on the image using the same formula as described in the bounding box method in order to define cell sizes.

A secondary method using *imregtform* involved placing a second ideal grid image on top of the binary mask of the image. For this method the diameters of the points in the ideal grid image are set to be 7 pixels.

3.2.3 Density based grouping

A Density based method is also applied in order to place the grid on top of the dot matrix in the image.

Two arrays are first created from the point array as $\{X_P, Y_P\} = P$ with each only containing point values for a single axis. The values in those arrays will be defined as $X_P = \{x_i\} \quad i \in I$, similarly for Y_P for the following formulas. Because these arrays inherently contain a side view of the dot grid, clusters could be formed which represent the cell center locations for the grid.

Due to the nature of the points having a small appearance rate and the possibility of containing noise, the number of possible clusters remained an unknown and can not be fixed. Because of this, traditional clustering methods like K-means can not be applied to separate the groups. The clumps are instead separated by using an iterative method on both of the arrays by using the average distance between neighboring points.

Firstly two arrays $C^x = \{x_0\}$ and $A^x = \{\}$ are defined and the values in X_P are iterated through using the following function so that $k \in I \cap \{i_0\}'$.

$$\begin{cases} C^x = C^x \frown x_k, & \text{if} \quad x_k - x_{k-1} < t \\ A^x = A^x \frown \mu_{C^x}; C^x = \{x_k\}, & \text{Otherwise} \end{cases} \tag{3.3}$$

The value t denotes a maximum distance threshold, usually set to $0.5 \cdot ||\mu_{\vec{G}}||$ for the best results. The average distance of these clumps is then calculated to get the grid cell size using

$$d^x = \frac{1}{|A^x|} \sum_{i=1}^{n-1} A_i^x - A_{i+1}^x \ , \tag{3.4}$$

and the average starting location of the grid is calculated as

$$l^x = \frac{1}{|A^x|} \sum_{i=1}^{n} A_i^x \bmod d^x \ . \tag{3.5}$$

In a situation where the average location of the grid should be near 0, which can be detected when the values of $(A_i^x \bmod d^x)$ in the formula 3.5 are distributed near 0 and d^x but not $\frac{d^x}{2}$. The point locations are moved by a quarter of the average distance and l^x is recalculated. $0.25d^x$ is then subtracted from the output to get the real average grid location. Otherwise the calculated average starting location value will be unreliable.

All of these functions are also applied to Y_P to find the Y axis equivalents. The point $\{l^x, l^y\}$ will then denote the starting point of the grid and the vector $\{d^x, d^y\}$ will be the diagonal of each cell in a grid over the entire image.

Next a 24×16 grid is defined with the same size parameters and iterated over the entire larger grid to find the best fit. The best fit will be where most of the data points are either inside of the grid or less than $\frac{||\{d^x, d^y\}||}{2}$ away from its outer boundary.

3.3 Point location

After the grid has been successfully placed over the dot matrix, each dot is isolated into its own individual cell, which will be notated as $C = \{c_{ij}\} \in \mathbb{R}^{mn}$ in the following formulas. Before the center values of the dots in each cell can be sampled, the dots have to be first located with better accuracy. In order to achieve this, three different methods are proposed and tested.

3.3.1 Averaged maximum

The first method used to locate the centers of the dots in each cell relies on the maximum values inside the cell.

A Gaussian blur filter is applied over the entire cell in order to remove smaller pepper noise or sharp bleed noise. The borders of the cell are padded with replicate padding in order reduce the influence of the filter on the result itself. With zero padding, constant value padding or no padding, the result would always end up near the center of the cell. The pixel location of the highest value in the image is then located and sampled, representing the point value of the dot. The average value of the background is also sampled as a reference.

3.3.2 Circle Hough Transform

Circle Hough Transform (CHT) is a commonly used method for detecting circular patterns with a fixed or range limited radii in image data [5]. One of its better features is the ability to detect circular patterns even when they are severely occluded or covered by noise [9].

It is applied using the Mathlab function called $imfindcircles$ with the intent of locating the circular dot locations in each cell. The specific type of CHT used is the Phase Coding variant as it performed better than other alternatives. [5, 37] If a circle is detected, its center is sampled as the dot value and the average background value is sampled as the reference. If no circular pattern is detected in the cell, the the dot value is equalized with the reference.

In practice the CHT method proves to be very unreliable when detecting circles with very small radii which in this case is often $2 - 6$ pixels. Because of this, the cells are re-sized to be $5\times$ larger than the original and a Gaussian blur is applied over the newly re-sized cells.

3.3.3 Kullback–Leibler divergence

The final method relies on the inherent shape of the dots, as they are very similar to a normal distribution. Because of noise often present in the cells, the average value can not be used to find the mean of the distribution. Instead, Kullback–Leibler Divergence (KLD) which measures the difference between two probability distributions, is used to find the best fit for a manually defined normal distribution [16]. The mean of the manually defined distribution can then be used as point center.

Two side view arrays of the cells contents are made as the average values of the cell in both the X and Y axis as $C^x = \{\mu_{C_j i}\}$, similarly for C^y. KLD is then applied to these arrays to measure the difference between the array and a normal distribution $Q(x)$. $Q(x)$ is then iterated over C^x as $Q^l = \{Q(x+l)\} \quad l \in \mathbb{R}^m$ until the smallest difference value is found. Because the mean of the iterating normal distribution is equal to the iterator, it is used as the exact center of the dot in an axis.

$$D_{KL}(C^x||Q^l) = \sum_i C_i^x \dot{\log}(\frac{C_i^x}{Q_i^l}) \tag{3.6}$$

Where in $\min\{D_{KL}(C^x||Q^l)\}$, l defines the location of the dot in the X axis. The same method is applied to C^y which will yield the Y coordinate of the dot. From the calculated dot location, a 3x3 grid is sampled and averaged to record the dot value within the cell. For the reference, the average value of the entire cell is also sampled.

3.3.4 Noisy point elimination

In order to eliminate noisy or unreliable points in the dataset, the positions of each calculated point are compared to their corresponding pair dots. The pair dots are located in the neighboring cell on the X axis. If their positions within their respective cells differ more than $0.36 \cdot ||\mu_{\vec{G}}||$, the sampled dot values of both are equalized with their respective background values. In order to minimize the amount of data being passed on, the average values of both the background and dot sample for each pair are used instead of the individual cell values.

4 Data manipulation

The previous methods produce a 3D matrix where the first two dimensions depict the dot location in the grid and the third dimension denotes whether the values is the dot value or its reference. Because each teat consists of 4 images, there matrices can be combined into a single 4D matrix where the fourth dimension describes the image in question. For the clustering based methods, the data can retain its 4 separate dimensions. In order to correctly label the results for the SVM, the data must first be re-sized into a single dimensional array for each test.

The matrix is reordered so that the values of each of the four images are next to each other so the fourth dimension can be removed. Next the third dimension is removed by making first value the dot values and the second one the reference values for each image. Finally the cells of the X axis are ordered together under every Y index. The matrix is re-sized in this order, because it allows for values in the matrix to be easily accessed.

4.1 Grid selection size

In order to reduce the amount of data going into training each SVM, which may cause overfitting issues. Different grid sizes of neighboring cells are used to generate the input data for the SVM's.

When selecting grid sizes, pair dots are always present in the selection process of each cell, This means cell count in the X axis can only be an even number. Because the data presented in the evaluated test files is grouped into pairs of two vertically, it is more convenient for the cell count in the Y axis to also increment in even numbers if the count goes over 2.

The smallest grid size being tested is a 2×1 which only contains the information for a given dot and its pair dot, resulting in an array of length 8. The other sizes are 6×4, 10×6 and 24×16, where the last one contains the entire grid.

4.2 Local Binary Pattern

Local Binary Pattern (LBP) [15] and Rotation-Invariant Local Binary Pattern (RI-LBP) [1, 22] are also applied to generate the data for each of the cells. LBP was selected both because of the methods simplicity and tolerance to illumination changes. The Rotation-Invariant version was mainly chosen in order to disregard the angle of noise in the data. As the sought after data is in a circular shape, RI-LBP can be better suited for distinguishing it from noise. [23, 24]

The method consists of applying the LBP filters on each of the cells individually in order to obtain information on the gradients within those cell. An array of pixel value occurrences is then constructed from the filtered cells contents.

In order to have a comparison with the paired cell, two arrays are created. The first array contains an average of the two occurrence value arrays and the other contains the mean absolute error between the two.

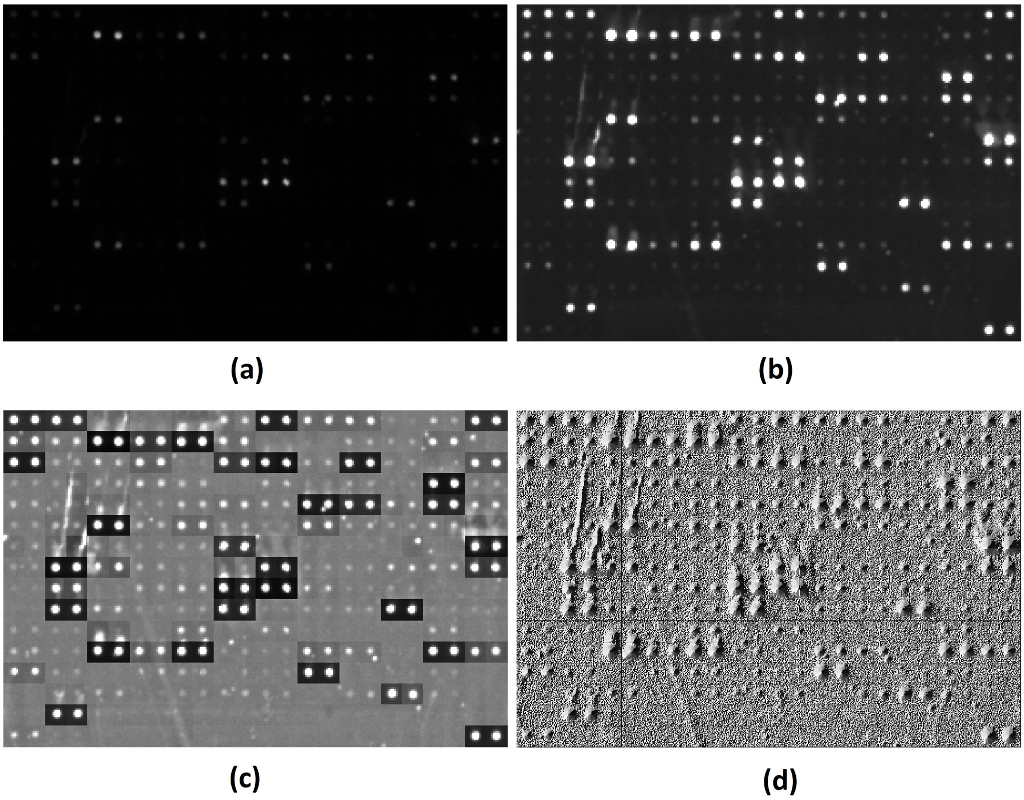

Figure 4.1: The data isolated from the input image in 8bit depth(a), illuminated(b), illumination applied separately for each cell(c) and with LBP(d)

In order not to overwhelm the SVM's with very long input arrays, only the values of the current cell are used to train the corresponding models. This means the SVM do not have any reference data from the surrounding cells when making their decisions.

With the intent of evaluating the LBP results, two other methods are also applied. One of those uses the raw cell value occurrences and the other applies Discrete Wavelet Transform and Singular Value Decomposition to illuminate the cell before sampling the occurrences [11].

5 Prediction

Each of the dots in the tests have different threshold values, all dependant on the values of the neighboring cells, the noise in the image, the index of the cell and the test image itself. Because of this manually assigning threshold values for each cell would be an overwhelmingly arduous task. Instead, both supervised and unsupervised learning techniques are applied to correctly predict the binary data of each test.

For the unsupervised methods, two well known clustering algorithms are applied. This is done so because the existing method uses a customized clustering algorithm in order to label the data.

Because of the nature of the data and the fact that there is not enough of it, more complex supervised learning methods like neural networks are not used to create a predictive model of this data-set. Instead, SVM are used as their problem with large data-sets can be avoided in this case. [28, 36]

5.1 Clustering

K-means [4] and Fuzzy C-means (FCM) [7] clustering are also applied on the dot and background reference values in order to create a baseline for the SVM based models. The clustering techniques are mainly applied in order to see if traditional unsupervised learning methods can produce acceptable results with this seemingly simple classification problem.

For the K-means clustering, the number of clusters is set to 2 and for each cell the dot and background reference values of all the cells in the grid are set as the two dimensions. Looking at the data distribution of those 2 dimensions implies that simple clustering might not be able draw the correct line between the 2 groups. In order to more uniformly distribute the results, the common logarithm of dot values is used instead of the raw dot values. K-medoids clustering

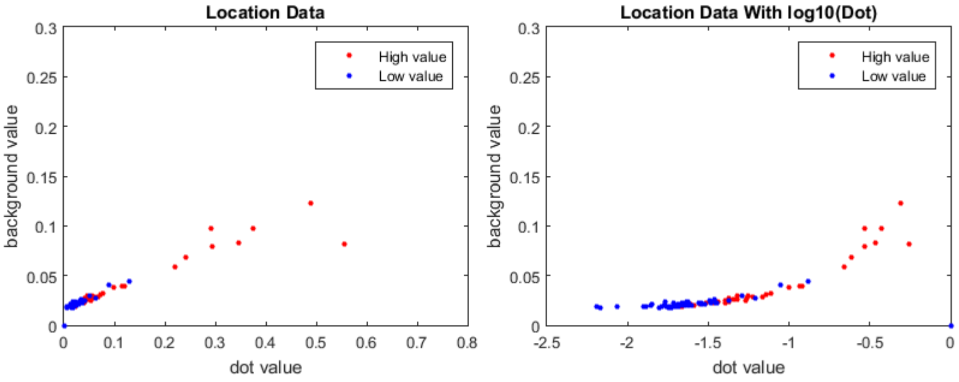

Figure 5.1: Data distribution for a single image with both the raw location data and with the common logarithm of dot values, The colors represent the actual data labels

is also applied on the dataset, but as the results are too similar to K-means it is not separately evaluated against the other methods. This is probably caused by the large presence of data points near the cluster centers which makes the results for K-medoids nearly identical to K-means.

FCM clustering is applied in a similar fashion with two groups. For the FCM, different ratios for the 2 clusters are also applied. Lastly, FCM is evaluated with 4 groups where the inputs are the dot and reference values for each of the 4 test images in 8 separate dimensions. This however produces results seemingly at random so the method is not evaluated against the other proposed techniques.

5.2 Support Vector Machine

In essence, a SVM constructs a hyperplane in a space where each of the features is a separate dimension. The data in that space is the labeled training data and the hyperplane separates the data into 2 groups based on their labels. [35]

Because of the high versatility of SVM's to use different types of input and output configurations to predict data, they are trained with varying features ranging from 8 to 1536 values in order to label a single cell. The different feature sets are chosen and compared against each other to select those with the best performance. The running time of the different models is generally not regarded when comparing the performances. This is mainly done so because even the methods with the worst run times completed their predictions in reasonable time frames of under 5 seconds per test. [35]

In order to better classify the results, different kernel functions are also applied for all of the models. [2] The main reason for this is to detect under- or over-fitting issues with the trained models.

For the application of the SVM model, the Mathlab functions called *svmtrain* and *svmclassify* are used to both train the model and predict its results. Because the model does not always converge with the default max iteration count of 15000, it is set to 100000. The method used to find the hyperplane is the Sequential Minimal Optimization [26]. Least squares and quadratic programming optimization are not used because neither of them improve accuracy and both increase the computational time significantly.

6 Experimental results

All of the experiments are conducted on 500 tests worth of data which span over a time period of 6 years. The prediction results from SVM's trained on the location data, LBP values, illuminated local and raw values are compared against both FCM and K-means clustering on the same test sets to see if a trained SVM has any significant advantage with this particular data-set. Several different amounts of neighboring cells are used as the input for the location data SVM's to test whether the model has come across an over-fitting problem when evaluating individual cells. The same SVM's are also trained using some of the more common kernel functions. The results from both the SVM and clustering models are also compared against the unedited results from the method that is already in place in Genorama® Genotyping SoftwareTM. The reason behind this is to see if the methods proposed in this paper can be used to improve upon the already existing system.

6.1 Grid alignment method evaluation

The three methods used to define the grids are all tested on a small sample size of 20 randomly chosen images and the results are evaluated manually. This is done so because there is no separate metric to automatically judge the quality of the grid placement.

Bounding boxes

The first method applies the average between a simple outer bounding box and a bounding box with 3 maximum and minimum values removed on each axis. This method produces fast and acceptable results on images that contain no noisy data outside of the grid and no sparsely populated edges. In cases where noise is present, the method often overextends the grid boundary, and where the edge data has very few detected points, the boundary cuts into the dot grid. In both cases the grid is inaccurately aligned resulting in incorrectly calculated cell dimensions and locations. This can be seen in figure 6.2a where both of these misalignments are present.

Image alignment

The second method uses the Mathlab function called *imregtform*, in order to align a perfect dot grid onto the image. In images where the dots are sparsely detected in the grid, *imregtform* fails to place the ideal grid on the image. However with densely packed dot grids the results are more promising. Because the majority of the test images contain a sparsely active grid of dots, this unfortunately means *imregtform* using the detected points fares even worse than the simple bounding box method.

Using the thresholded images instead of the detected points provides very similar results in images with low noise and the difference only becomes apparent in images with larger anomaly

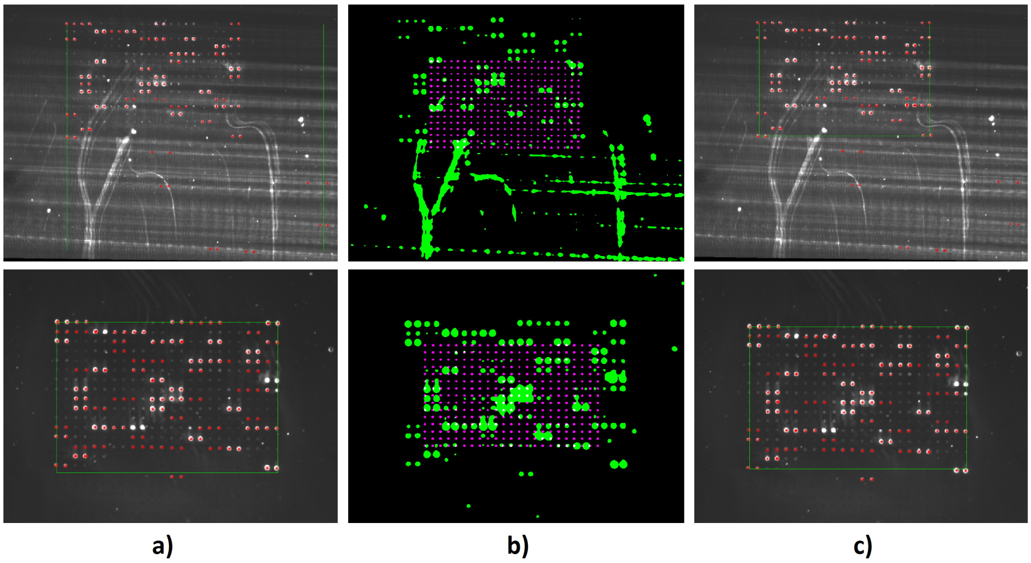

Figure 6.1: Grid alignment attempts on two noisy test images using bounding boxes(a), *imregtform*(b) and the density based grouping method(c). The images on the left and right have the bounding boxes defined in green and all of the detected points after image cleanup in red. Bounding boxes should intersect the edge most points in the data grid. The middle images have an ideal grid mask defined in magenta and the original image mask in green.

noise. In both cases as the test images often resemble less than 40% of the ideal image, a reliable transformation can not be calculated.

Density based grouping

The last method which is based on the X and Y point densities, very often produces the correct number of groups. Cases where 1 group is missing or an extra one is generated is luckily very low, but the results in those cases are greatly improved by iterating a grid over the proposed data locations.

The results provided by the density based grouping method greatly surpass what either the bounding box or image alignment methods could produce with this dataset. Because of this, all of the further operations use the method for the grid placement.

6.2 Point location method evaluation

All of the point location methods are manually evaluated over 20 randomly chosen test images. Because each methods accuracy is very apparent and visually comparable, no error calculations are needed to select the best method.

Averaged maximum

The averaged maximum method accurately locates the dots in cells where no high value noise is present as seen in Fig. 6.2a. Unfortunately a substantial amount of cells contain high value pepper noise, which makes this method unreliable. Applying different types of blur on the cells does reduce the effect of pepper noise, but this has little effect in increasing the accuracy of this method.

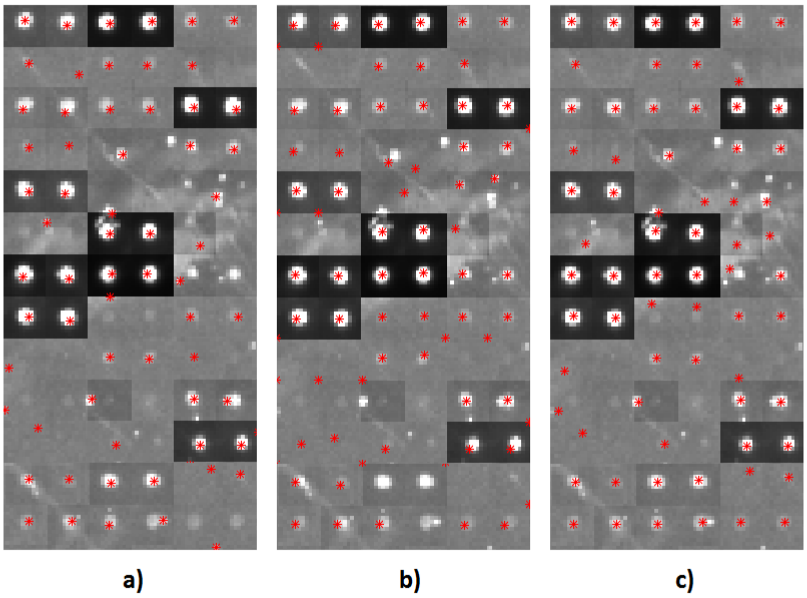

Figure 6.2: Three point location allocation methods on 78 different cells. The locations are found using the averaged maximum(a), CHT(b) and KLD(c).

This method also suffers greatly from the small section size as the method used for padding the edges greatly influenced the result. Zero padding and constant padding causes the detected location of the dot to drift towards the center while extended edge padding causes the opposite to occur.

Hough transform template matching

The results provided by the Hough transform based circle detection are unfortunately also very unreliable. The results are easily influenced by noise as the method often can not distinguish vibrant pepper noise with a smaller radius from actual data dots. Very often the method also misplaces the center of a detected dot near its edge, invalidating the center sampling results. The method also proves to be the most resource intensive as going over and detecting dots in a single image takes upwards of about 5 seconds on an i7-4790@3.6GHz. For comparison, the other applied methods take less than a second to work through an entire test.

Kullback–Leibler divergence

Compared to the two previous methods, KLD provides the most accurate results. The method is unaffected by sharp pepper noise and in some cases even manages to locate the data in cells overlapped by large anomaly type noise. Something the other 2 methods fail at but where KLD excels is locating the exact centers of the dots. This ensures the detected value is not in fact noise but actual data. A great example of this is the 4th image on the bottom row in Fig. 6.2 where the averaged maximum method sampled noise, CHT failed to even locate the circle and KLD found the center perfectly.

Table 6.1: Average accuracy of clustering techniques evaluated using 10 fold cross-validation

Clustering on location data						
Method Measure	K-means	K-means with log10(high value)	FCM 2:1	FCM 1:1	FCM 1:2	FCM 1:10
Accuracy	75.24%	69.25%	75.35%	74.73%	73.60%	71.73%
Precision	24.57%	23.33%	24.80%	24.64%	24.13%	23.51%
Recall	6.74%	18.16%	6.60%	7.91%	10.03%	13.36%

6.3 Clustering results

K-means and Fuzzy C-means (FCM) clustering methods are applied on the point value data in order to compare their outputs against the customised clustering method used in the existing software. It is also used as a baseline for all of the trained SVM results. The clustering methods are evaluated on the entire 500 test dataset and the measures are calculated as the average values for each of the images separately.

For both clustering methods, the data is clustered into 2 groups, where each group denoted either the presence or absence of a dot in one of the images in a test. For the K-means clustering, a second input method is evaluated where the common logarithm of high values is used instead of the raw variant. FCM is evaluated using different ratios for assigning the 2 binary values. In the table the ratio is represented as the average background value over the dot value.

In order to properly compare the methods, other measures called precision and recall are used alongside accuracy. Precision is the rate of true positive values over the predicted positives and recall is the same but over the actual positive values. Because type I and type II errors have similar severity within these tests, Accuracy is still considered the most important measure.

Neither of the tested clustering methods can surpass even the worst results provided by SVM based models that use the same input data for evaluation in table 6.3. In both cases, K-means and FCM produce very small precision and recall rates, meaning the overall true positive rate is very low. Along with that, because the recall rates are very small, the models are producing very high type II error rates. By increasing the ratio to favor dot values, the recall of the model steadily improve but the overall accuracy decreases. the opposite is true the other way around, but that leads to the model no longer being a predictive system.

6.4 Support Vector Machine results

All of the methods that use SVM's to predict the output are evaluated using 10-fold crossvalidation. The 500 tests are split into 10 groups of 50, 9 groups are used for training and 1 for evaluation. This is done for each of the 10 groups in order to calculate the average accuracy of each model. In most cases only the accuracy is evaluated, because both types of errors carry equal value.

Location data

The SVM based method is trained and evaluated on the entire dataset of located points using 4 different types of input grid sizes and 3 different kernel functions. The amount of data fed into the SVM varies in order to see if the model is over-fitting. The data amount variations consist of different sized girds around each dot with the 4 grid sizes used being 2×1, 6×4, 10×6

Table 6.3: Average accuracy of different location based models evaluated using 10 fold cross-validation.

Accuracy of models trained on location data			
Kernel Function / Grid Size	Linear	Quadratic	3rd Order Polynomial
2×1	89.33%	94.84%	96.64%
6×4	97.68%	98.35%	97.79%
10×6	98.20%	98.56%	94.84%
24×16	98.79%	98.73%	97.19%

Table 6.5: Average accuracy of only the SVM's evaluated using 10 fold cross-validation, non-changing labels were not evaluated.

Accuracy of SVM's on location data			
Kernel Function / Grid Size	Linear	Quadratic	3rd Order Polynomial
2×1	75.99%	88.40%	92.28%
6×4	94.77%	96.28%	95.04%
10×6	95.95%	96.76%	88.43%
24×16	97.27%	97.04%	93.67%

and the last version using the entire grid values as the input to calculate every single datapoint. The 3 kernel functions used are linear, quadratic and 3rd order polynomial kernels. They vary in complexity because they are used to evaluate if the model is experiencing under-fitting.

Two methods are used to calculate the accuracy of all of the trained models. The first method directly compares the binary results from the model to the binary representation of the expected results. The proportion of similar results is deemed as the accuracy. The second method considers the fact that several of the binary values don't change between the tests so those are discarded in the evaluation process. The accuracy in the latter case is purely calculated on the binary values that change between different tests. This is done in order to get a better understanding of the models performances.

As expected, increasing the input grid size has diminishing returns on the accuracy. What is not expected however, is how increasing the grid size well beyond the local area of each point does not result in obvious over-fitting. Out of all the methods tested, using location data from every point in the grid provides the best accuracy in predicting the expected values. The model trained on the entire grid as the input is the only one that performs better with a simple linear kernel. The two methods using smaller selections of local points perform better with quadratic kernels. Something to note is how the neighborless 2×1 grid as an outlier experiences severe under-fitting, enough so that increasing the kernel function complexity has a significant impact on increasing the performance of the trained model. Even though the accuracy of the 2×1 grid is very low, the amounts of type I and type II errors remain relatively equal. The same is true for all of the other methods using SVM's trained on location data.

Table 6.7: Average accuracy of different cell based models evaluated using 10 fold cross-validation

Accuracy of models trained on cell data				
	LBP	RI-LBP	Raw Cell Values	Cell Illumination Enhancement
Total accuracy	98.58%	92.12%	83.49%	98.57%
Accuracy over changing values	96.80%	82.26%	62.86%	93.12%

Cell data

Local Binary Pattern (LBP) values and Rotation-Invariant Local Binary Pattern (RI-LBP) values of each cell are calculated and used as an input for the SVM's. This is done in order to compare and evaluate the results of the SVM's trained on the point values themselves. Two other methods are also used which do not use the LBP filter output to generate values for each cell. The first of these methods uses the raw high 8 bits of the values in each cell while the other applies Discrete Wavelet Transform and Singular Value Decomposition before converting them into 8 bit values. Both of these are used as a control in order to assess the impact of applying LBP on the dataset. The reason for using illumination enhancement on the cells is to counteract the effects of background anomaly type noise. In order to calculate the accuracy of all of the methods, the same two accuracy based evaluation techniques are applied here.

The results from LBP are very promising as they provide the second highest accuracy of all the tested methods, only beaten by the local values of each point. What is unexpected is how poorly RI-LBP performs when compared to standard LBP and simple illumination enhancement. The worst results are provided by the unaltered local cell values. The reason for the low accuracy in that model is mostly caused by background anomaly type noise. This is because large uniform value changes in the background displace dot thresholds in individual cells, making the predictions comparable to random guessing. The small difference between the LBP and illumination enhanced cell data is expected because the illumination enhanced version doesn't hold any gradient direction information. The lack of gradient direction makes distinguishing noise like bleed or large anomalies in the data much harder for the illumination method.

6.5 Evaluation against existing method

Out of the 500 tests worth of data, 20 tests were selected to be evaluated by the method already in place in Genorama® Genotyping SoftwareTM. Out of these 20, 10 were selected at random, 5 were manually chosen as being visually the worst to evaluate by hand and 5 were manually chosen as the best to evaluate by hand. The the best and worst were selected purely by the amount of noise visually visible in the data images and not by the performance of any model. This means there are in total 6440 data points from 10 random tests, 3220 data points from 5 of the best and similar for the worst tests that can be used to compare the accuracy of the proposed methods against the existing method. The methods chosen to be compared against the existing one are the best location based models for each grid size, the LBP and RI-LBP variants, as well as the simple cell illumination enhancement method. Because the clustering attempts yield very poor results, they are not evaluated against the existing method.

Table 6.9: The best of the proposed models evaluated against the existing model on the 10 manually selected and 10 random test samples. The training sets for the proposed models included all tests part from the one being evaluated

Proposed models evaluated against existing method			
Evaluation Sample Model / Grid Size / Kernel	Best Set Accuracy	Worst Set Accuracy	Random Set Accuracy
Existing Method	96.18%	96.50%	94.71%
Location / 24 × 16 / Linear	99.47%	98.87%	97.00%
Location / 10 × 6 / Quadratic	98.97%	98.53%	96.92%
Location / 6 × 4 / Quadratic	99.05%	97.95%	96.89%
Location / 2 × 1 / 3rd Order Polynomial	95.97%	94.76%	94.97%
LBP / 2 × 1 / Linear	98.84%	99.00%	96.82%
RI-LBP / 2 × 1 / Linear	92.16%	91.79%	91.12%
Cell Illumination / 2 × 1 / Linear	98.61%	97.68%	96.38%

Among all of the best models tested, only the location based model with a 3rd order polynomial kernel, and the RI-LBP model with a linear kernel perform worse than the existing method. All of the other methods show a significant increases in accuracy with the 2 best being the location based model with a full grid input and the standard LBP with its local cell as the input.

Comparing the confusion matrices of the different classifiers, there is actually very little difference. All of the models have very small differences between their false positive and false negative counts. The main difference between the existing method and the location based linear model is the false positive rate which, over the random selection, are 5.06% and 2.22% respectively. This means that the existing model produces 3 times as many type I errors as type II, while the proposed methods all have a ratio closer to $3 : 2$, which can be seen on Table 6.11. Even with that being the case, the 2 proposed methods that perform the best, both still produce type II error counts significantly lower than that of the existing methods error count.

Table 6.11: Test results for best models for random set. The training set for the proposed models included all tests part from the random set

Test results for best models				
Count Method	True Positive	True Negative	False Positive	False Negative
Existing Method	1513	5682	303	102
Location Based Method	1520	5852	133	95
LBP	1515	5843	142	100

7 Conclusion and Future Work

Several methods were developed for both automatically aligning the grid on the image and locating specific dots in cells. They have all been evaluated against each other in terms of accuracy by visual comparisons. The best methods managed to correctly and reliably extract the information from even the noisiest images in the database. This shows the information extraction process from the APEX images can in fact be easily automated.

The second part of the work saw the application of SVM's with varying configurations evaluated against the existing method currently in place in the Genorama® Genotyping SoftwareTM. Several of the proposed methods produce accuracies that are significantly better than the ones proposed by the existing software. Two of the best methods which each employed very different techniques for the input managed to achieve accuracies exceeding 98.5% and 98.7%, making them comparable alternatives for the automatic evaluation system.

Both of the proposed methods can be further improved to incorporate better noise detection higher accuracy in the predictions. The dot detection process could incorporate circularity detection [29] in order to eliminate noise which could have an effect on the output. The best prediction methods could be combined together with a voting system which could detect questionable cells. The grid alignment method itself could also be changed to incorporate varying grid sizes and multiple grids in a single image for larger test images. With this the method could be used for different PCR related applications or even for the automated inspection of ball solder joints [17].

8 Acknowledgments

I would like to thank my supervisor Assoc. Prof. Gholamreza Anbarjafari (Shahab) for guiding and encouraging me throughout the thesis process. I would also like to thank the other members of iCV for giving me ideas when I got stuck.

Most importantly, I would like to thank Asper Biotech for giving me access to their data, without which this work would have been impossible.

Bibliography

[1] Timo Ahonen, Jiří Matas, Chu He, and Matti Pietikäinen. Rotation invariant image description with local binary pattern histogram fourier features. *Image analysis*, pages 61–70, 2009.

[2] Shun-ichi Amari and Si Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

[3] Gholamreza Anbarjafari. Face recognition using color local binary pattern from mutually independent color channels. *EURASIP Journal on Image and Video Processing*, 2013(1):6, 2013.

[4] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[5] Tim J Atherton and Darren J Kerbyson. Size invariant circle detection. *Image and Vision computing*, 17(11):795–803, 1999.

[6] Elena Bernardis. Finding dots in microscopic images. 2011.

[7] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[8] Melvin Cohen and Godfried T Toussaint. On the detection of structures in noisy pictures. *Pattern Recognition*, 9(2):95–98, 1977.

[9] E Roy Davies. *Machine vision: theory, algorithms, practicalities*. Elsevier, 2004.

[10] Hasan Demirel, Gholamreza Anbarjafari, and Mohammad N Sabet Jahromi. Image equalization based on singular value decomposition. In *Computer and Information Sciences, 2008. ISCIS'08. 23rd International Symposium on*, pages 1–5. IEEE, 2008.

[11] Hasan Demirel, Cagri Ozcinar, and Gholamreza Anbarjafari. Satellite image contrast enhancement using discrete wavelet transform and singular value decomposition. *IEEE Geoscience and remote sensing letters*, 7(2):333–337, 2010.

[12] Zhenhua Guo, Lei Zhang, and David Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Transactions on Image Processing*, 19(6):1657–1663, 2010.

[13] Juha Hirvonen and Pasi Kallio. Scale and rotation invariant two view microgripper detection that uses a planar pattern. *IFAC Proceedings Volumes*, 46(5):414–422, 2013.

[14] Ching Yu Austin Huang, Joel Studebaker, Anton Yuryev, Jianping Huang, Kathryn E Scott, Jennifer Kuebler, Shobha Varde, Steven Alfisi, Craig A Gelfand, Mark Pohl, et al. Auto-validation of fluorescent primer extension genotyping assay using signal clustering and neural networks. *BMC bioinformatics*, 5(1):36, 2004.

[15] Di Huang, Caifeng Shan, Mohsen Ardabilian, Yunhong Wang, and Liming Chen. Local binary patterns and its application to facial image analysis: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(6):765–781, 2011.

[16] James M Joyce. Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*, pages 720–722. Springer, 2011.

[17] Kuk Won Ko, Young Jun Roh, Hyung Suck Cho, and Hyung Cheol Kimn. A neural network approach to the inspection of ball grid array solder joints on printed circuit boards. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 5, pages 233–238. IEEE, 2000.

[18] Ants Kurg, Neeme Tõnisson, Ioannis Georgiou, John Shumaker, Jeff Tollett, and Andres Metspalu. Arrayed primer extension: solid-phase four-color dna resequencing and mutation detection technology. *Genetic testing*, 4(1):1–7, 2000.

[19] Roland Linder, Tereza Richards, and Mathias Wagner. Microarray data classified by artificial neural networks. *Microarrays: Volume 2: Applications and Data Analysis*, pages 345–372, 2007.

[20] Genorama Ltd. *Genorama® Genotyping SoftwareTM 4.5*, 2002-2009.

[21] Eneli Oitmaa. *Development of arrayed primer extension microarray assays for molecular diagnostic applications*. PhD thesis, 2013.

[22] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Gray scale and rotation invariant texture classification with local binary patterns. In *European Conference on Computer Vision*, pages 404–420. Springer, 2000.

[23] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[24] Matti Pietikäinen. Image analysis with local binary patterns. *Image Analysis*, pages 6–9, 2005.

[25] Michael C Pirrung, Richard V Connors, Amy L Odenbaugh, Michael P Montague-Smith, Nathan G Walcott, and Jeff J Tollett. The arrayed primer extension method for dna microchip analysis. molecular computation of satisfaction problems. *Journal of the American Chemical Society*, 122(9):1873–1882, 2000.

[26] John C Platt. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pages 185–208, 1999.

[27] Maido Remm, Kaarel Krjutškov, and res Metspalu. Primer design for large-scale multiplex pcr and arrayed primer extension. In *PCR Technology: Current Innovations, Third Edition*, pages 199–208. CRC Press, 2013.

[28] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[29] Richard Shen, Jian-Bing Fan, Derek Campbell, Weihua Chang, Jing Chen, Dennis Doucet, Jo Yeakley, Marina Bibikova, Eliza Wickham Garcia, Celeste McBride, et al. High-throughput snp genotyping on universal bead arrays. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 573(1):70–82, 2005.

[30] Priit Tomson. Dna mikrokiipidel kasutatavate oligote kvaliteeti mõjutavad parameetrid ja meetodid nende disainiks. pages 20–21, 2005.

[31] Neeme Tõnisson. *Mutation detection by primer extension on oligonucleotide microarrays*. Tartu: Tartu University Press, 2002.

[32] Neeme Tõnisson, Jana Zernant, Ants Kurg, Hendrik Pavel, Georg Slavin, Hanno Roomere, Aune Meiel, Pierre Hainaut, and Andres Metspalu. Evaluating the arrayed primer extension resequencing assay of tp53 tumor suppressor gene. *Proceedings of the National Academy of Sciences*, 99(8):5503–5508, 2002.

[33] David C Walley, Ben W Tripp, Young C Song, Keith R Walley, and Scott J Tebbutt. Macgt: multi-dimensional automated clustering genotyping tool for analysis of microarray-based mini-sequencing data. *Bioinformatics*, 22(9):1147–1149, 2006.

[34] Junchen Wang, Etsuko Kobayashi, and Ichiro Sakuma. Coarse-to-fine dot array marker detection with accurate edge localization for stereo visual tracking. *Biomedical Signal Processing and Control*, 15:49–59, 2015.

[35] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for svms. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 647–653. MIT Press, 2000.

[36] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svms with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315. ACM, 2003.

[37] HK Yuen, John Princen, John Illingworth, and Josef Kittler. Comparative study of hough transform methods for circle finding. *Image and vision computing*, 8(1):71–77, 1990.

Non-exclusive licence to reproduce thesis and make thesis public

I, Rain Eric Haamer

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   (a) reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   (b) make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   "Automated Data Extraction and Analysis for Arrayed Primer Extension Images"

   supervised by Assoc. Prof. Gholamreza Anbarjafari

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **17.05.2017**