DNA Sequence Analyses and Error Correction: A Major Qualifying Project

A Term Major Qualifying Project Report

Submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

Advisors: Professor Patrick Flaherty

Submitted By:

Wai Phyo Maung

Maris Pepo

Jared Erb

Alec Rebh

Date: Thursday, April 30, 2014

TABLE OF CONTENT

AUTHORSHIP

Abstract Alec
Chapter 1 Maris, Jared, Alec and Wai
Chapter 2
- section 2.1 Alec
- Section 2.2 Wai
- section 2.3 Maris
- section 2.4 Jared

Chapter 3 Maris, Jared, Alec and Wai
Chapter 4
- section 4.1 Maris
- Section 4.2 Maris
- Section 4.3 Maris and Wai
- Section 4.4 Maris
- Section 4.5 Maris

Chapter 5
- Section 5.1 Maris
- Section 5.2 Maris
- Section 5.3 Maris

Chapter 6
- Section 6.1 Wai
- Section 6.2 Alec
- Section 6.3 Jared
- Section 6.4 Maris
- Section 6.5 Alec
- Section 6.6 Alec

Chapter 7
- Section 7.1 Wai
- Section 7.2 Alec
- Section 7.3 Jared

Chapter 8 Maris and Jared

ACKNOWLEDGEMENTS

In the completion of this Major Qualifying Project we would like to thank our advisor Professor Patrick Flaherty for his guidance. Additionally we would like to thank Jason Rosenman for helping create a virtual workspace for us to code in. We would like to also give a special thanks to Moly Miranda from Stanford University for providing us with output sequencing data from a illumina machine.

ABSTRACT

Advances in error correction for next generation sequencing have not matched increases in data production and as a result, the quality of the generated nucleotide sequences has suffered. The purpose of this project was to develop a processing pipeline which would remove errors from intensity data for a faster and more accurate analysis. The method employed to achieve these goals was to redesign the algorithm used to correct for bleaching and phasing to capture a greater number of misidentified bases. Two pipelines were created, pipeline 1 (illumina) and pipeline 2 (Oracle), it was determined that pipeline 2 out preformed pipeline one in terms of accuracy. But pipeline 1 was determined to be faster in processing time and thus the main question is asked do you sacrifice time for efficiency?

CHAPTER 1-INTRODUCTION

Next generation DNA sequencing technologies allow for a much faster processing of genetic information than more traditional methods. These new methods lay heavy emphasis on micro-imaging and fast paced data processing. The leader in this new age technology, with their Sequencing by Synthesis (SBS) technique, is the California based company Illumina®. Using their SBS technology Illumina is capable of analyzing multiple genomes in a week's time, producing many terabytes of data in one sequencing run (illumina). The most commonly established method prior to SBS is gel electrophoresis which sequences by separating and categorizing fragments of DNA by charge and size, often taking many days to acquire a relatively small amount of data (Sinville, 1714). In just 4 years, Illumina has been able to drastically increase the throughput of their SBS systems, resulting in a 1000-fold increase in data per sequencing run through optimization and refinement alone (illumina). As next-generation sequencing develops, it becomes clear that there is a need for faster genomic analysis. In areas like oncology, a fast and accurate analysis of gene expression signatures can lead to the most effective therapy (illumina clinical). Unfortunately, the error rate of incorrectly identified nucleotides during the sequencing process, while low, has remained relatively consistent. As data output increases, a consistent error rate results in many more incorrect bases, which can negatively impact the results of the sequencing.

Currently Illumina deals with DNA sequencing by fragmenting the sequence and adhering the pieces to a chemically engineered plate known as a flow cell. Each nucleotide in the sequence is then phosphorised and imaged in series to produce a collection of maps

indicating nucleotide location at different levels. These maps are then corrected for multiple sources of error (illumina products).

The team hypothesizes that a large portion of the error is created during two specific steps in the sequencing process used to correct for disparities in color intensities. The first step accounts for the possibility of fluorescent color mixing and the ultimate misreading of a nucleotide. The second step corrects the “phasing” or skipping of base calls in a sequence. Through optimization of the image analysis algorithms and statistical analysis of the fluorescent nucleotide intensities the team will lower this error rate, resulting in a more precise genome sequence.

The project has two main goals: the experimental goal and the engineering goal. Illumina's sequencing algorithm has some flaws with identifying the clusters correctly giving an error percentage of .25%. The first part, which is the experimental portion, of the project will deal with identifying the source of error. The team will look at the raw DNA sequence data and try to pinpoint the cause of bases being misidentified. In the second part, the engineering portion of the project, the team is going to produce an image processing algorithm that clearly corrects and explains the error. By doing this, the team hopes to improve the performance of the state-of-the-art software in aspects including the reduction of processing time, reduced error rate, portability and robustness.

The overall goal of this project is to create a more efficient pipeline for the post processing of next generation sequencing machines. This will be accomplished through the use of python language to create a modified algorithm. The current pipeline will be investigated and duplicated to determine the causes of implementing each process within the pipeline. The

next generation image processing pipeline has two main processes by which base calling and quality score is determined. The first process which will be looked at will be the color matrix, which is used to correct for the variance in fluorescents. The other process, which will also be the focus of the project, will be the errors in phase change, which help to align the base images. A close examination will be done to determine how each color matrix and phase changes are determined. These issues will be investigated to illuminate the cause of the .25% error seen in the processing. After examining each process, modifications will be made and run through the raw data provided by sequencing machine. A closer examination will be done to determine the most efficient changes, specifically in the color matrix and the phase parameter estimation, which are more clearly explained in later chapters. A newly generated pipeline will be produced and compared to the old pipeline. This comparison will result in an increase quality control score for the sequences data and more accurate base calls. This accomplishment will result in creating a more efficient pipeline for the processing of the next generation machine.

CHAPTER 2-LITERATURE REVIEW

2.1 Illumina Background

Since the very beginning of genomic research, capillary based sequencing methods have dominated the field. As sequencing progresses and ultimately plays a larger role in disease diagnostics it is increasingly imperative that both the cost and run time of sequencing technology decreases. Conventional DNA sequencing methods have plateaued in terms of throughput, leading to a need for innovative sequencing approaches. The effort to produce novel sequencing techniques has been attempted by many companies. Most of these new companies are based on a cyclic-array system utilizing repetitions of image-based data collection. Array generation and biochemistry often differ between next generation processes but ultimately, the work flow of each company involves DNA fragmentation, duplication and imaging (Ji, 2008). While these new methods have numerous advantages, including higher parallelism and lower reagent costs, they are also known to have error rates of up to ten fold that of traditional Sanger methods. Ewing and Green propose that a large contributor to the error of next generation sequencing machines is substitutions in the bases being called (Ewing & Green, 1998). These higher error rates necessitate estimation for the likelihood of an incorrect base call. In a separate paper, Ewing et all test the use of a modified phred algorithm to produce an accurate quality score (Ewing et. all, 1998). This score is intended to calculate the reliability of each next generation sequence. Because of the statistical nature of this algorithm, it is applicable across several different next generation sequencing methods.

Illumina, the leader in these next generation sequencing processes, uses parallel oligonucleotide adaptation to duplicate DNA fragments and produce cluster maps which can be

imaged to reveal sequences of nucleotides (Macevicz, 1998). The work flow for Illumina's processes is revealed through documentation on their website.

Fragmented DNA is adhered to a flow cell, which is a proprietary surface chemically designed to hold the DNA strands upright. The fragments are then amplified into clusters of the same strand through a bridging and cloning process.

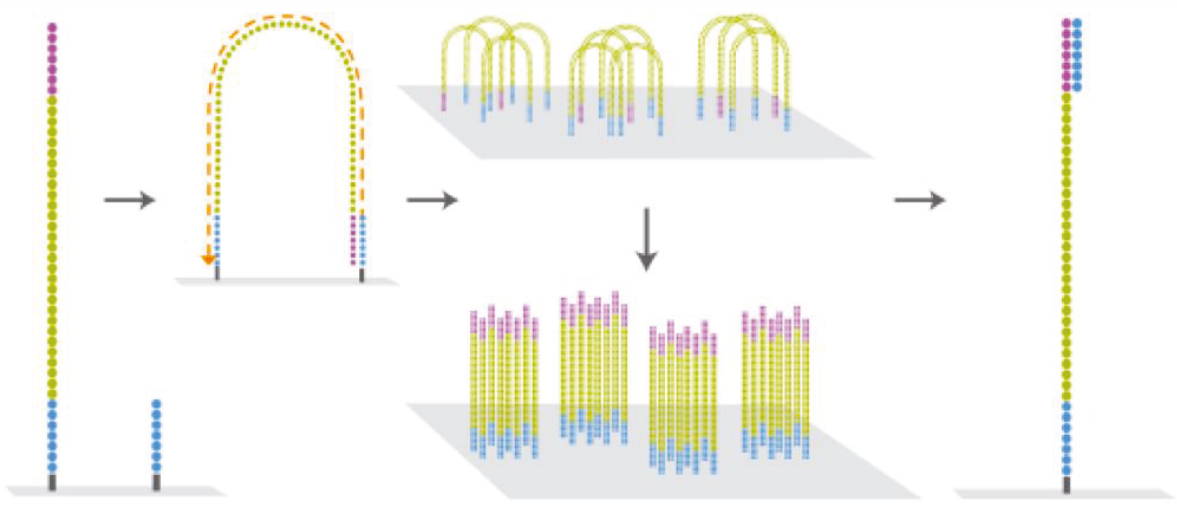

Figure 1: A graphical representation of Illumina's cluster generation (Goldeberg, 2011)

This allows for a greater intensity when fluoresced. To achieve this fluorescence, single fluorescein-labeled deoxyribonucleoside triphosphate species are added in cycles to produce color specific fluorescence for each nucleotide in a read. Image processing software is then able to identify the nucleotide of each cluster in each cycle (Aksyonov, 2005). For every cycle, the flow cell is scanned in 6 parts and those 6 parts are divided into 8 tiles. Each tile runs procedurally through a series of steps, resulting in millions of computing processes.

All sequencing begins with the template generation, a process that defines the locations of every cluster on a flow cell. Once a template has been generated, processing for all other cycles can begin. For each cluster position an intensity value is extracted using image processing

software. The images are processed on four separate channels, meaning each nucleotide is identified by a separate color in its own image file. The laplacian transform of these scanned image files is used to create an array of intensities, indicating an order of bases. These intensities are stored in a cluster intensity file. Because a different color is used to identify each nucleotide, a method for correcting crosstalk, or the blending of colors is needed to reduce error. A color matrix is produced using the color intensity files to correct for these errors. Each component of the matrix represents the amount of crosstalk from each nucleotide observed in every other nucleotide's channel, resulting in a four by four matrix. To counteract the error produced by substituted, deleted and added bases, Illumina uses a phasing estimation technique. The phasing estimation assumes that in each cycle, a certain number of DNA strands incorrectly identify one base in the sequence, whether through the addition, substitution or deletion of one nucleotide. An equation is calculated to account for these "phased" molecules. The phase corrected intensities are then used to call the bases for each DNA fragment. The quality score that indicates the probability of an incorrect base call is then calculated. The final step in Illumina's process is the alignment of all the fragmented base reads using a control sequence (RTA Theory of Operations, 2011).

2.2 Information on the freeIbis

The most recent improvement to the illumina's next generation pipelines is a base calling software named FreeIbis. This new software has been shown to "out preform the previous version of software in terms of sequence accuracy". (Renaud, Kircher, Stenzel, & Kelso, 2013) The accuracy of the base calling is determined based on the ability of the correct base calls to accurately correspond to its respective genome. This was then compared to speed of

the run and the percentage of mapped sequences. The table below shows that freeIbis not only out preforms other software in terms of run time, but also in percent accuracy of correctly matching a sequence to the genome. The table below shows 0.16% increase in accuracy over Ibis and a 1.55% increase in improvement over illumina's sequencing pipeline.

Table 1: Accuracy of each bascaller on an Illumina GAIIx dataset (Renaud et al., 2013)

Basecaller	Training time	Calling time	Mapped (%)[a]	Edit distance
Bustard			583 348 201 (83.93%)	1.379
naiveBayesCall	591 h	658 h	578 957 145 (83.34%)	1.496
AYB	394 h		593 183 967 (85.52%)	1.076
Ibis	19.4 h	13.2 h	592 929 953 (85.31%)	1.167
freeIbis	21.3 h	12.2 h	594 095 219 (85.48%)	1.145

The human sequences were mapped to the hg19 version of the human genome. The number of mapped sequences and the average number of mismatches for those were tallied for each method. Time trials were conducted on a machine with 74 GB of RAM and using 8 of the 12 Intel Xeon cores running at 2.27 GHz. [a]Percentage relative to sequences assigned to the read group of interest.

Illumina's pipeline currently produces a way of determining the accuracy of the base calling called the quality score of the run. This demonstrates how accurate the pipeline produced the right base calls, a higher quality score means that the bases are more accurately matched. The freeIbis was run through the quality score and a predicted line was determined for all 4 types of bases. This was done by producing a root mean square. As can be seen in figure 1 a side by side plot shows the higher accuracy of freeIbis as compared to bustard, which is named for illumina's pipeline process. The graphs show the significant improvement in calling of each base and the uniformity of all of the bases. (Renaud, Kircher, Stenzel, & Kelso, 2013)

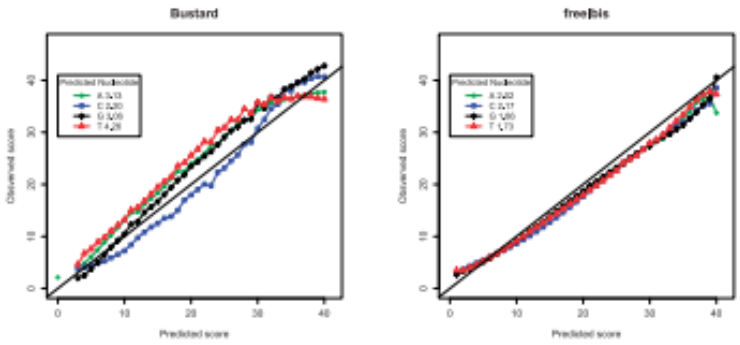

Figure 2: Bustard vs. Freelbis RMS line (Renaud et al., 2013)

2.3 Real World Application

Illumina's next generation sequencing is utilized in many fields as a form of research and diagnostics. These machines are used in such field like genetic diseases research, forensics, and microbiology and cancer research. Through the use of these machines different strains of infections can be differentiated based on the genetic variations. In the case of tracking influenza H7N9 in china researchers used Illumina next generation machines to identify patients with this infection and show variation in strains. The machines allowed researchers in Jiangsu Provincial Center for disease control and prevention to positively identify H7N9 in humans and animal, specifically chickens, to confirm a strong probability that the pathogen was spread from chickens to humans. (Illumina) Using the same machines researchers were able to identify patients which were infected with multiple influenza strains, which allowed for a good understand as to "the mechanisms of viral assortment from which new strains emerge" (Illumina, 2013). These machines have become crucial in understanding and preventing the spread of pathogens within our environment. These machines have made it possible to obtain more accurate and faster sequencing of DNA, which in terms of identifying pathogens means containing the virus quickly and identifying the root of the virus to help in obtaining the proper treatment. This means that viruses will not be allowed to spread through to the masses and

become pandemics or epidemics causing mass infections and deaths. Thus it is crucial that the Illumina Next Generation Machines become as accurate as possible.

Currently these machines have been accurate in terms of the chemistry component and thus it would be highly beneficial to look at the digital aspect which has yet to be perfected and optimized to its full capacity. Small variations in genetic testing could mean a wrong diagnosis and in terms of genetics this could be a life or death test. More and more diagnosticians have made genetic testing a commonality. It has been shown that genetic testing can determine the likely hood of obtaining a disease later on in life. Due to the dramatically lower cost of illumina's machines and the rapid turnaround time it has made it possible for patients with rare disease to obtain genetic testing which will help not only the patients, but also the families obtain answers as to what may be causing strange symptoms. In the case of the Sukins family who had a child which was showing abnormal signs of a disorder, it wasn't till testing was done that they were able to find that their son had a rare genetic disorder called Angelman syndrome, with this they can help find ways of treating the symptoms of the disorder. (Kolata) These revolutionary diagnosis help in predicting patients with the likely hood to obtain a disease in the future, which has made it possible to come up with treatment plans and help patients live longer. Illumina's Next Generation Machines are crucial in medicine today helping to increase the life span of many patients.

2.4 Problems and issues

In principle bases can be called straight from intensity files but there are several complicating factors that must be dealt with, cross-talk, phasing and dimming being of particular importance.

Cross-talk is the recording of light from a single fluorophore in multiple channels. This occurs because, although they are chosen to be distinguishable, the fluorophores' emission spectra overlap. There is not a one-to-one correspondence between channels and FLNs and the relationship between the emission of each fluorophore and the intensity observed in each channel needs to be ascertained and corrected for.

Phasing refers to the deterioration in relationship between sequencing cycle and sequence position as the cluster loses coherence: on a given cycle, FLNs may be attaching to different positions on different molecules within the cluster. There are many possible explanations for phasing: for example, a FLN might have a defective reversible terminator element leading to the attachment of two FLNs to a molecule on a single cycle, allowing the molecule to get ahead in the sequencing process ('pre-phased'), or the cleaving of the reversible terminator might fail for a cycle so the molecule lags behind when the element is finally removed ('post-phased'). A further possible cause of post-phased molecules is the chemistry not running to completion, resulting in either no FLN being attached that cycle or cleaving failure as previously mentioned. Finally, molecules within a cluster gradually stop contributing to the total signal, possible causes being laser damage to the individual molecules or problems reversing the terminator element, and this leads to a decrease (dimming) in the overall emission observed from each cluster in later cycles of sequencing.

Bleaching is the final factor that influences the error of the data. Bleaching occurs when the fluorescent dye, which is used to stain the bases, degrades over the time. This error occurs in later cycles because they are the last to be scanned. The result of this degradation is a lower

numerical result for all bases in that cycle. This can cause a misread in bases and result in higher error towards the end of readings.

The cross-talk is a consequence of the physics of fluorophore excitation and methods for estimating it have already been developed for dye-terminated capillary electrophoresis sequencing platforms. Phasing and dimming are more specific to NGS methods, the Illumina platform in particular, and have been approached in a variety of ways. The Illumina base calling software (Bustard) (Kao, Stevens, Song,2009) assumes a constant rate of post-phasing and pre-phasing for all cycles that allows the phasing at each position of the sequence to depend on several of the neighboring bases .

CHAPTER 3-PROJECT STRATEGY

When initially presented with the problem, the team was unaware of certain project constraints and produced an open ended client statement. The initial client statement was "Identify and correct the underlying cause of sequencing error of cluster mapping methods of current benchtop sequencing devices. This correction will be achieved through the engineering of a software pipeline which will show improvement over the current methods."

The team's main objectives were created from the initial client statement. These objectives created the framework for the design and the constraints of the project. First the team needs to test the hypothesis that the major cause of the error in the current software pipeline is due to image processing once the hypothesis has been verified the team aims to design a new software pipeline that has a lower error rate than the current system that is faster and more robust in its design. The team also has a stretch goal of creating the pipeline to have a portable version that will operate on a standard consumer laptop.

The objectives were then compared using a pairwise comparison chart as shown below in table 2. The chart shows which of the objectives are most crucial to the project and as can be seen the objective to create a pipeline which resulted in a lower error rate for the base pairs came out on top. Using the objectives an objective tree, as seen in figure 3, was also created to see how these objectives can be properly implemented into the project and help to determine which objective would require the most amount of work.

Table 2: Objectives Pairwise Comparison Chart

	Lower error rate	Software robustness	Faster processing time	Software portability	Total score
1.Lower error rate	X	1	1	1	3
2.Software robustness	0	X	1	1	2
3.Faster processing time	0	0	X	1	1
4.Software portability	0	0	0	X	0

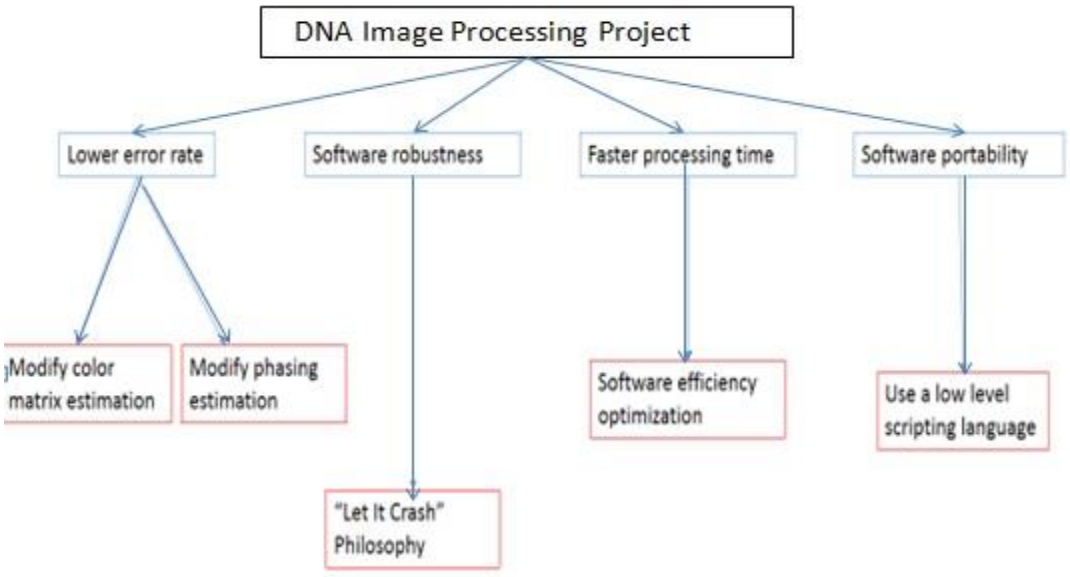

Figure 3: Objective Tree

The team has many constraints that were considered while designing the pipeline. The first is system performance as the team aims to be faster than the current process. Currently it requires 40 hours of processing to handle a 2 lane sequencing run of the illumina HiSeq scanner. The second is the need to conserve disk space. The goal is to have no more than 1.5X overage in data processing. This is to prevent the system from producing data faster than it is being processed. The largest constraint the team followed is the need reduce the error rate from .25% to .025%

After meeting with the client again and considering the constraints the team revised the client statement. The revised client statement is “Develop an image processing software pipeline that is more accurate than the current pipeline from Illumina. The software should be at least as fast as the current state-of-the art. The improved pipeline should lead to a more complete and thorough understanding of the error that occurs in the image data and solutions to decrease the impact to those sources of error by at least 10x on the variant calling algorithm. “In the revised statement, detailed requirements for the software relative to the current state-of-the-art were stated. The notion of image processing was introduced as well since the team will be solving the problem mainly through image processing. Also, an important constraint was added in the form of 10x improvement in the software accuracy.

The team had numerous financial concerns when developing the new pipeline. The first was a storage media. The Illumina HiSeq machine develops close to 4Tb of data on any run. This storage media was purchased from an online distributor. The second was the HiSeq run itself. A lab donated this so the team incurred no cost on this portion. The third is the use of a research cluster to perform the processing. The team had access to a machine at no cost. The fourth the

team investigated the ability to use the Amazon Web Service EC2 Cloud computing solution. The development on the cloud would allow startup labs using this software to run sequences without having to purchase a research cluster to do it.

The team applied numerous analytical tools to the research and development of the new pipeline. One major method used was the ceiling method. This process looks at the problem as a whole and breaks it down into pieces. These pieces are then examined one by one to determine which part has the greatest effect on the error of the code. The team took each piece and improved the accuracy and speed and then retested the over all error rate of the pipeline. This process allows the problem to be broken down and managed by the team. This is shown in the Work Breakdown Structure below. The team will also use best practices, including memory management and Big O efficiencies when developing the new code. This allowed the team to create a design evaluation matrix, shown below, to evaluate the different designs.

The design evaluation matrix below depicts the process that was taken in order to determine which method would be most efficient for the project. The first method that was chosen to be analyzed is the ceiling method which was described below. This method was compared in its ability to input the correct file, which in this case would be the .cif files and its ability to output high quality scores, which means that the number of accurate base calls is at a much higher percentage. The other method that was used for comparison was the guess and check method, which takes different statistical function and implements it into the code to see if it would give a higher quality score. The error robustness sees what errors occur within the current code and finds a method to correct for those which would give a higher quality score. Then the last method would be complete overhaul, this would involve completely eliminating

the code that was most recently found and create a new code which would have the same input and output, but a high quality score would be produced. Looking at table 3 it was determined that the best course of action for this project would be to implement the ceiling method in analyzing the code obtained for the Illumina next generation machine.

Table 3: Design Evaluation Matrix

	Input Cif Files	Output Fastq Files	High Quality Scores	Totals
Ceiling Method	4	5	5	14
Guess and Check	4	3	4	11
Error Robustness	4	3	4	11
Complete Overhaul	2	1	2	5

The Gantt chart below shows the timeline with which we hope to complete this project. The chart shows a general approach of gathering background research within the first week of the project. This included researching articles and the most up to date codes which have dealt with problems in illumina's next generation machines. The next 7 weeks of the project will be composed of placing a ceiling method of analysis to the current code that illumina's machines implement. By focusing in on the color matrix and the phase shift of the code meticulous analysis will be done to determine flaws within the current method. The next quarter of the project will deal with determining C functions which will perform similar tasks as the current method but with more accuracy. The most effective change made will be determined by placing a quality score on the changed code. The last part of the project will be finishing all of the writing aspects and determine the validation of our changed code and preparing for our final presentation.

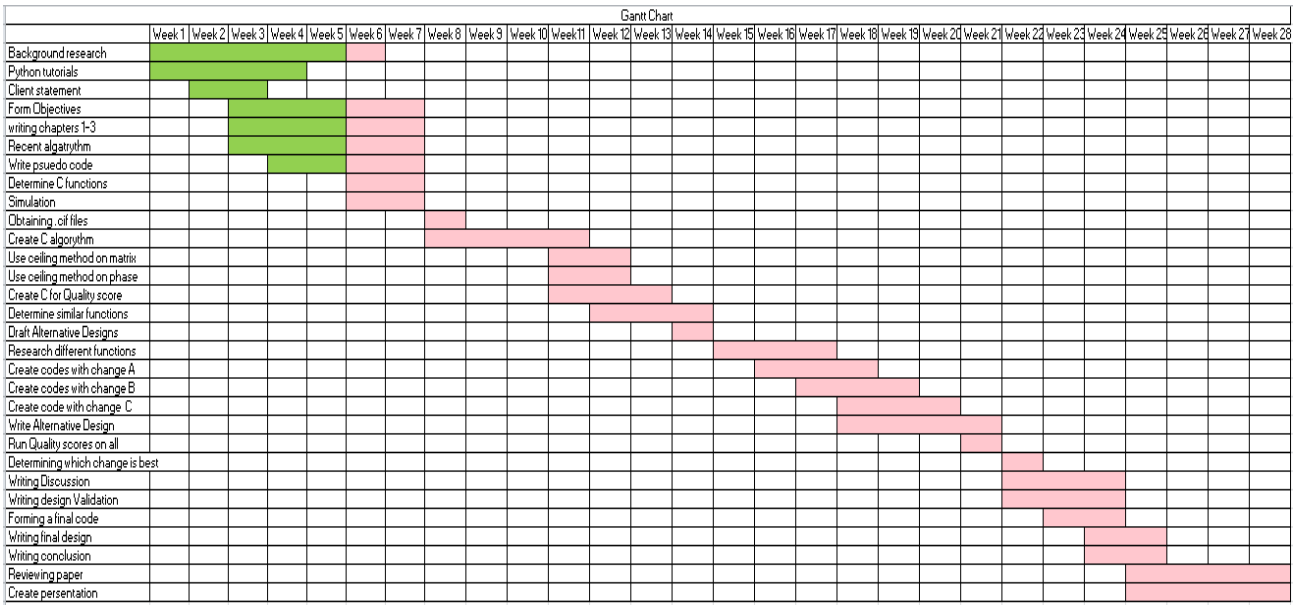

Figure 4: Gantt Chart

In the work Breakdown Structure seen below, the basic layout of the DNA Image Processing MQP project. This project will be separated into two parts, one in which currently implemented code is analyzed and another part which involves creating a new more efficient code. When analyzing the recent code, a ceiling method, which was described in a previous part, will be implemented. This will involve a specific focus in two main parts the color matrix and the phase shift. Both of these will be run through quality scores to see which parts have the most effect on the accuracy of the base calls. This will help the group to focus on which part of the code needs changes. This leads to the next part of the code, which involves creating a new more efficient code. This will involve coming up with ideally three changes to the code which will also be run through quality scores to determine which changed code preforms the best. This will lead to a complete project goal of gaining a 10X improved base calling algorithm compared to that of the most current algorithm.

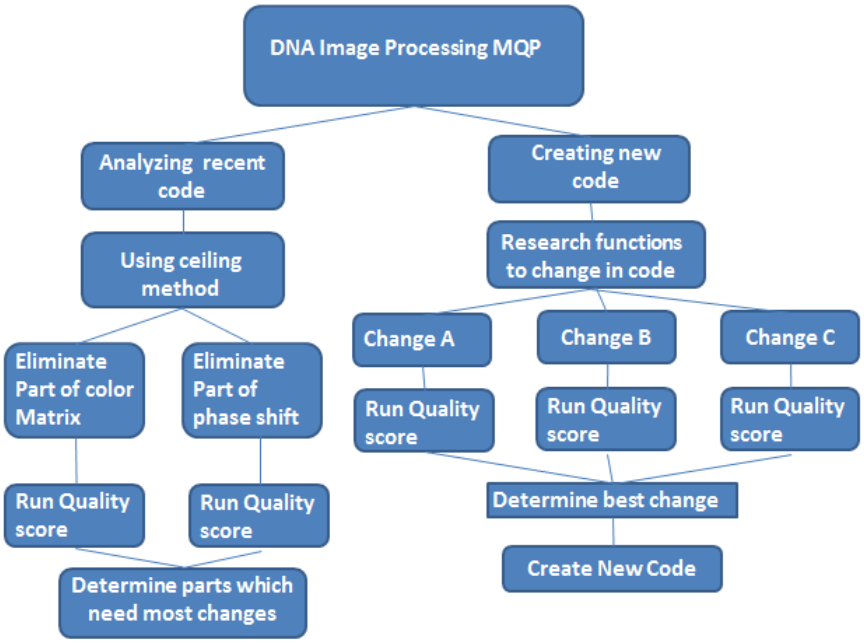

Figure 5: Work Breakdown Structure

CHAPTER 4-ALTERNATIVE DESIGNS

4.1 Needs Analysis

After discussing normal DNA analysis practices, the team came to an understanding about specific requirements needed in order to make a feasible improvement to Illumina's first generation sequencing pipeline. The first requirement is the pipeline must be able to be processed using the normal speed and memory of an average laptop. These specifications include a 64GHz core 2 due processor and a 256 GB RAM memory. This makes it easily transferable and able to be processed in in any laptop device.

Ideally the pipeline would be 1.5 times faster, meaning it would take less than 40 hours for 4TB of data to run. Even with no improvement to the error rate of the pipeline gaining a faster run time would make it more efficient and thus make it a better analysis tool for the instrument. Another ideal for this project would be creating a pipeline which is 10 times more effective in reading bases then the current pipeline. Although the main goal of the project is just to make a more affective pipeline, it would be ideal to make a high performance more effective pipeline which runs on laptop.

4.2 Functions

The following functions were developed by the team.

- Read and input .cif information

  The pipeline should be able to analysis and convert .cif files from the Illumina machine to numerical values to be analyzed and modified, then converted back into .cif files to accurately call bases.

- Output .fastq files

  Fastq files are the standard text-based format for storing biological sequences and their corresponding quality scores, thus the pipeline must be able to output this type of file for better comparisons.

- Higher percentage of accurate base calls

  The pipeline must be able to call a higher number of accurate bases then that compared to the Illumina pipeline.

- Create a simulation algorithm to output .cif files

  By creating simulation data which duplicates the output of the Illumina machine a set of control data can be created for analysis of the pipeline and can also be used to model different types of sequences.

4.3 Conceptual Designs

4.3.1 Chastity determiner

One alternative method for obtaining a phase and prophase ratio to correct for color phasing is to utilize the chastity equation as seen below. The chastity is originally used as a way of filtering data which falls below a threshold.

$$chastity = \frac{(Max_{value} - second_{Max})}{Max_{value}}$$

Equation 1: Chastity

Using this method the chastity of each cluster for each cycle based on the set of bases. Each base would be compared to the other bases for that cluster at that cycle. If the chastity value passes a set threshold for that base then a count increases by one. A count is established

for each base through all clusters for that cycle. The count is then set over the total runs of chastity to get a ratio for each cycle. Cycles (1:end-1) will then be averaged out to create a phase ratio. Cycles (2:end) will again be averaged to obtain a prophase ratio. These ratios will then be put through the rest of the pipeline and analyzed to determine if a higher number of basses were called.

This method has fall backs associated in terms of setting a threshold for the chastity and since it is a value based on purity of the data, this threshold would have to be at a range which filters all impure and phased data, but keeps clean data with other forms of noise. Although this may result in a pipeline which does not improve the system, it would show improvement in the speed of the run and thus would be a viable choice for creating pipeline 2.

4.3.2 Autoregressive function

Although the reverse engineering of Illumina's sequencing method and the team's oracle method were fully functional, the pipelines had some room to improve in terms of base call accuracy. The team came up with an alternative method of finding the phasing and prephasing parameters by trying to fit the data in an autoregressive model and estimating the values. Autoregressive model seemed to have similar characteristics with phasing since there is linear association between lagged observations.

Matlab has a built-in function,namely "EstMdl", that can use maximum likelihood to estimate the parameters of an ARIMA(p,D,q) model where

p=Positive integer indicating the degree of the nonseasonal autoregressive polynomial.

D=Nonnegative integer indicating the degree of nonseasonal integration in the linear time series

q=Positive integer indicating the degree of the nonseasonal moving average polynomial

The team used the ARIMA(1,0,0) model assuming the degree of AR polynomial to be 1, the degree of integration to be 0, and the degree of MA polynomial to be 0, for each data set of cycles for a single cluster.

Phasing parameter was calculated by dividing the data into arrays (each array consisting of data from all cycles for a single cluster), fitting the arrays into the ARIMA(1,0,0) model and estimating the parameter. The same process is done to calculate the prephasing parameters with the arrays in the opposite direction.

It was found that the ARIMA model estimates the phasing/prephasing values more accurately than the team's previous methods. However, the estimation script is run in MATLB , and takes a large runtime. At this moment, the team only reached to the point of getting the phase/prephase parameters, but not to the point of data correction with those values and base calling.

4.4 Decision

The final design was determined using the design matrix seen in the figure below. As seen, the winning design was established to be the autoregressive due to the high mathematical property that is involved. The Autoregressive method would involve no estimation or preexisting knowledge of the data being implemented into the pipeline. Thus based on a range of 0-5 the autoregressive function received a 5 for obtaining a lower error

percentage. Being that a lower error rate is the top objective for the project, it was chosen as the implemented design.

Design Matrix			
	oracle phase and pre-phase	Autoregressive	Chastity Determiner
Lower Error (O)	3	5	1
Software robustness(O)	2	1	1
faster processing (O)	2	1	2
Software portability (O)	5	5	5
Data Storage (C)	4	5	5
Total	16	17	14

Figure 6: Design Matrix for Alternative Designs

The oracle phase and pre-phase method would involve previous knowledge of phase and prophase ratio for the data which would not normally be known in a biological lab, making it a less useful method then the other two methods being considered. Although it ranked high on the design matrix, this method is not realistic in a real world application.

The chastity determiner method was ranked lowest on the design matrix and this is primarily due to the fact that a set threshold has to be determined. This would mean that if the threshold is not set to an optimal configuration then the error could potentially be higher than

that seen in Illumina's data. Due to an unknown threshold, this method ranked lowest in possibility of providing a lower error rating.

After many discussions and using the design matrix the final choice was to choose the autoregressive design as pipeline 2. This decision was made mainly due to the fact that using the autoregressive method, the probability of the pipeline being more effective and resulting in a lower error rating was the highest. Being that the main goal of this project is to create a functioning pipeline which out preforms Illumina's, it was decided that the autoregressive method would be most efficient in meeting this goal.

4.5 Preliminary Tests

A preliminary code was developed to determine the accuracy of the Autoregressive function created in Matlab. This function would input simulated clean data and simulated data which includes a phasing component of about 0.10. If the code functioned optimally then the phase values would match those created in the simulation data. This function would then separate the data by bases, then by clusters and run through the autoregressive forward through all cycles.

```
ARIMA(1,0,0) Model:
--------------------
Conditional Probability Distribution: Gaussian

                               Standard          t
 Parameter       Value           Error       Statistic
-----------   -----------   ------------   -----------
 Constant         360.62         120.517         2.99229
    AR{1}     0.00939809       0.0971345       0.0967534
 Variance         316628         49625.1          6.3804
```

Figure 7: output Autoregressive Model for clean data

The image above shows the results from an input clean data set which showed no signs of phasing or pre-phasing. As can be seen the AR(1) value which corresponds to the phasing that is occurring in that base for that cluster of data. Since this result shows a value relatively close to 0, it can be assumed that the code created would not modify the data when no phasing has occurred. This same test was run for a simulated data set which showed a phase and pre-phase value of 0.1.

```
ARIMA(1,0,0) Model:
--------------------
Conditional Probability Distribution: Gaussian

                              Standard          t
 Parameter        Value          Error       Statistic
-----------   -----------   ------------   -----------
 Constant          355.41        266.412         1.33406
    AR{1}        0.207242       0.127137         1.63006
 Variance           287736         174997         1.64424
```

Figure 8: output Autoregressive Model for phase 0.1 and pre-phase 0.1

The figure above shows that the autoregressive function accounts for both the phasing and pre-phasing values seen in the data. Thus due to the fact that both phasing and pre-phasing values were 0.1, the resultant would be assumed to be 0.2. In the figure above the value of AR(1) very closely matches the expected value of 0.2. This preliminary test reveals the feasibility of using the autoregressive method in order to determine the occurrence of both phasing and pre-phasing from the sequencing data. From this point the data can then be corrected for phasing and pre-phasing resulting in a high percent accuracy for the base caller.

CHAPTER 5-DESIGN VERIFICATION

There methods were tested to determine efficiency of each method on the simulated data which was created. The first method was based on Illumina's technical document, which gave a step by step approach on recreating the Illumina pipeline. The second approach was to use an oracle approach that took into account the known phasing variable created using the simulator. The last method uses an autoregressive method in order to make future predictions based on current values. Tests for Error percentage and run times were created for both method 1 and method 2, named pipeline 1 and pipeline 2 accordingly. Method 3 was run for initial phasing and pre-phasing results, since a pattern was seen in the data and based on run time of the method no further tests need to be done.

5.1 Percent Error Results

The first test was to determine the percent error that occurred after running each pipeline. This was done by having a data set with known sequences and applying different phasing and pre-phasing components to that data. The simulation data would then be placed through the pipeline and a base caller would call each maximum value and compare the called sequence to the original clean sequence. Table 4 and Table 5 show the Percent error resulting from pipeline 1 and pipeline 2. Looking at the percent error results from pipeline 1, no trend or correlation can be seen by changing phasing and pre-phasing value of the data. Only phasing and pre-phasing for a small amount of simulation data was run because no initial tend was seen meaning that the phasing and pre-phasing components of the data had no effect on this pipeline. This pipeline seems to result in randomized data which demonstrates that this method was very unreliable and ineffective in terms of producing a low percent error in the data.

Simulation Table for Pipeline 1: Percent Error							
			Run 1 (%)	Run 2 (%)	Run 3 (%)	Run 4 (%)	Run 5 (%)
			cycles 3_12	cycles 1_20	cycles 20_50	cycles 3_33	cycles 50_70
clean	clean	clean					
decay rate 0.02	phase rate 0.05	pre-phase rate 0.05	53.905	12.081	73.396	53.905	81.959
		pre-phase rate 0.1	56.456	3.278	86.09	56.456	89.803
		pre-phase rate 0.15	75.121	30.866	42.047	75.121	86.623
		pre-phase rate 0.20	93.305	46.269	80.663	93.305	93.34
		pre-phase rate 0.25	55.936	69.394	84.217	55.936	94.766
		pre-phase rate 0.30	55.936	69.394	84.217	55.936	94.766
		pre-phase rate 0.35	73.119	73.409	77.658	73.119	88.485
		pre-phase rate 0.40	31.75	74.103	24.942	31.75	74.069
	phase rate 0.1	pre-phase rate 0.05	40.806	98.365	14.073	40.806	79.757
		pre-phase rate 0.1	25.671	82.887	30.868	25.671	87.319
		pre-phase rate 0.15	82.67	35.58	86.203	82.67	95.189
		pre-phase rate 0.20	75.366	98.436	7.265	75.366	98.104
		pre-phase rate 0.25	89.153	45.738	15.359	89.153	66.766
		pre-phase rate 0.30	89.153	45.738	15.359	89.153	66.766
		pre-phase rate 0.35	13.762	90.095	79.044	13.762	86.836
		pre-phase rate 0.40	16.434	40.017	59.578	16.434	83.904
	phase rate 0.15	pre-phase rate 0.05	73.11	82.964	65.123	73.11	98.398
		pre-phase rate 0.1	13.891	95.423	87.886	13.891	88.034
		pre-phase rate 0.15	70.46	70.656	78.43	70.46	93.539
		pre-phase rate 0.20	90.171	83.583	90.774	90.171	73.4
		pre-phase rate 0.25	57.629	71.751	74.784	57.629	89.06
		pre-phase rate 0.30	57.629	71.751	74.784	57.629	89.06
		pre-phase rate 0.35	68.109	68.716	82.543	68.109	79.769

		pre-phase rate 0.40	76.199	39.28	76.815	76.199	81.292
	phase rate 0.20	pre-phase rate 0.05	86.349	43.696	67.775	86.349	90.94
		pre-phase rate 0.1	91.382	67.764	74.476	91.382	90.531
		pre-phase rate 0.15	70.748	75.658	87.054	70.748	90.802
		pre-phase rate 0.20	91.493	78.845	85.715	91.493	91.74
		pre-phase rate 0.25	42.48	51.01	90.156	42.48	91.266
		pre-phase rate 0.30	42.48	51.01	90.156	42.48	91.266
		pre-phase rate 0.35	48.581	80.151	81.25	48.581	73.138
		pre-phase rate 0.40	77.328	79.456	63.278	77.328	83.978

Table 4: Pipeline 1 Percent Error Rate

Pipeline 2 was also run for percent error. As can be seen in table 2 below the method of iterating through the data and using a known phasing and pre-phasing variable produced a percent error which shows a positive correlation between phasing and percent error. With higher phasing and pre-phasing variables the percent error seems to increase proportionally to the increase in phasing and pre-phasing variables. One example can be seen when phasing was set to 0.05 and pre-phasing was set to 0.05 the resultant was about 0.001 percent. Now when we compare that value to the one seen in the simulation data using phasing set at 0.1 and pre-phasing set at 0.029 this means that there is about a 30X increase in error with ever 2X increase in phasing. The data shows a linear trend displaying a higher percentage of errors with higher phasing and pre-phasing values. The cycle values did not have any effect on this method due to the fact that this method accounts for all cycles, thus an average is taken and would not affect the resultant error. These results make sense with the simulation data input into the pipeline and have shown relatively good results.

Simulation Table for Pipeline 2: Percent Error							
			Run 1 (%)	Run 2 (%)	Run 3 (%)	Run 4 (%)	Run 5 (%)
			cycles 3_12	cycles 1_20	cycles 20_50	cycles 3_33	cycles 50_70
clean	clean	clean	0	0	0	0	0
decay rate 0.02	phase rate 0.05	pre-phase rate 0.05	0.001	0.001	0.001	0.001	0.001
		pre-phase rate 0.1	0.02	0.02	0.02	0.02	0.02
		pre-phase rate 0.15	0.382	0.382	0.382	0.382	0.382
		pre-phase rate 0.20	1.545	1.545	1.545	1.545	1.545
		pre-phase rate 0.25	3.399	3.399	3.399	3.399	3.399
		pre-phase rate 0.30	3.399	3.399	3.399	3.399	3.399
		pre-phase rate 0.35	8.267	8.267	8.267	8.267	8.267
		pre-phase rate 0.40	10.701	10.701	10.701	10.701	10.701
	phase rate 0.1	pre-phase rate 0.05	0.029	0.029	0.029	0.029	0.029
		pre-phase rate 0.1	0.108	0.108	0.108	0.108	0.108
		pre-phase rate 0.15	0.653	0.653	0.653	0.653	0.653
		pre-phase rate 0.20	1.867	1.867	1.867	1.867	1.867
		pre-phase rate 0.25	3.859	3.859	3.859	3.859	3.859
		pre-phase rate 0.30	3.859	3.859	3.859	3.859	3.859
		pre-phase rate 0.35	8.686	8.686	8.686	8.686	8.686
		pre-phase rate 0.40	11.687	11.687	11.687	11.687	11.687
	phase rate 0.15	pre-phase rate 0.05	0.373	0.373	0.373	0.373	0.373
		pre-phase rate 0.1	0.606	0.606	0.606	0.606	0.606
		pre-phase rate 0.15	1.27	1.27	1.27	1.27	1.27
		pre-phase rate 0.20	2.7	2.7	2.7	2.7	2.7
		pre-phase rate 0.25	4.875	4.875	4.875	4.875	4.875
		pre-phase rate 0.30	4.875	4.875	4.875	4.875	4.875
		pre-phase rate 0.35	9.973	9.973	9.973	9.973	9.973
		pre-phase rate 0.40	13.072	13.072	13.072	13.072	13.072

	phase rate 0.20	pre-phase rate 0.05	1.514	1.514	1.514	1.514	1.514
		pre-phase rate 0.1	1.767	1.767	1.767	1.767	1.767
		pre-phase rate 0.15	2.692	2.692	2.692	2.692	2.692
		pre-phase rate 0.20	4.131	4.131	4.131	4.131	4.131
		pre-phase rate 0.25	6.41	6.41	6.41	6.41	6.41
		pre-phase rate 0.30	6.41	6.41	6.41	6.41	6.41
		pre-phase rate 0.35	12.156	12.156	12.156	12.156	12.156
		pre-phase rate 0.40	15.24	15.24	15.24	15.24	15.24

Table 5: Pipeline 2 Percent Error Results

In order to get a full idea of the trend seen in each pipeline a graphical representation was created for each pipeline based on cycles 3 through 12. Figure 9 below shows the results of the data seen in tables 4 and 5. When looking at the results from pipeline 1, the randomization is very apparent, no trend is visible or apparent and the scale of the graph is shown from 10 to 100, displaying the wide variation in the data. The disarray of the data is apparent when you compare it to the graphical representation of pipeline 2, which shows a linear, even a slight exponential trend in data. The scale for this graph is seen between 0 to 15 percent revealing a lot less error in terms of the general numbers and in terms of the difference in ranges. This comparison shows the higher efficiency of pipeline 1 as compared to pipeline 2.

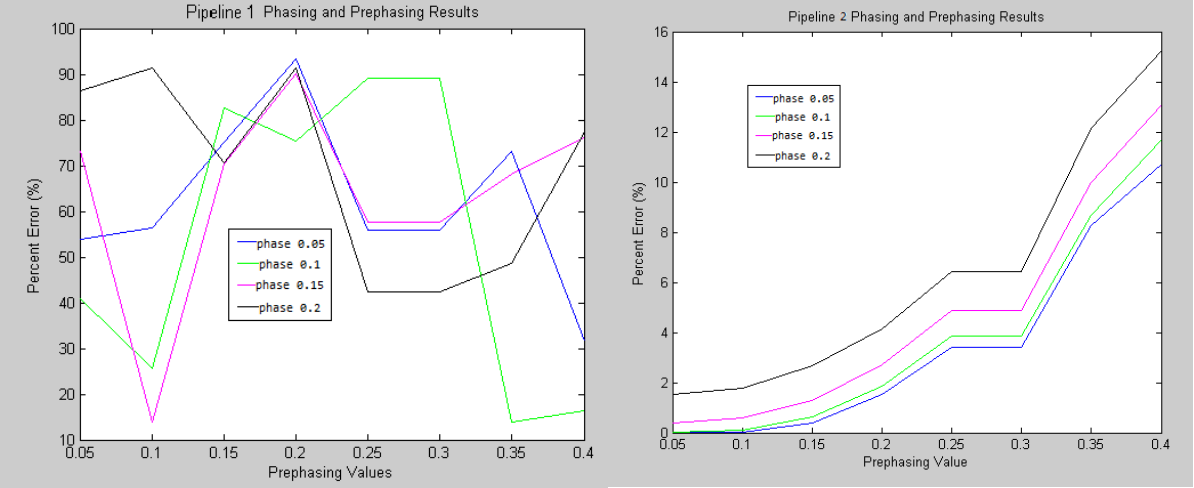

Figure 9: Comparison of Phasing for Pipeline 1 and 2

5.2 Run Time Results

Along with the percent error, a run time was also measured as a means of finding the most efficient method. The data from the run times of pipeline 1 and pipeline 2 can be seen below in table 6 and table 7. Table 6 shows the run time results from pipeline 1, although the run times have some variation, it is very miniscule. This means even with higher error in the data the amount of processing time will be relatively the same. Thus this allows for researchers to know the exact time the data will need to process which allows for turnover time of data to be more efficient.

Simulation Table for Pipeline 1: Run Times							
			Run 1 (sec)	Run 2 (sec)	Run 3 (sec)	Run 4 (sec)	Run 5 (sec)
			cycles 3_12	cycles 1_20	cycles 20_50	cycles 3_33	cycles 50_70
clean	clean	clean	3.36	3.12	3.15	3.12	3.18
decay rate 0.02	phase rate 0.05	pre-phase rate 0.05	3.36	3.12	3.15	3.12	3.18
		pre-phase rate 0.1	3.33	3.37	3.26	3.38	3.18
		pre-phase rate 0.15	3.25	3.32	3.13	3.24	3.22
		pre-phase rate 0.20	3.14	3.18	3.14	3.13	3.22

		pre-phase rate 0.25	3.1	3.08	3.11	3.13	3.11
		pre-phase rate 0.30	3.13	3.13	3.15	3.15	3.12
		pre-phase rate 0.35	3.13	3.14	3.1	3.13	3.16
		pre-phase rate 0.40	3.15	3.11	3.12	3.17	3.05
	phase rate 0.1	pre-phase rate 0.05	3.12	3.11	3.08	3.09	3.07
		pre-phase rate 0.1	3.07	3.1	3.11	3.11	3.15
		pre-phase rate 0.15	3.17	3.07	3.11	3.1	3.1
		pre-phase rate 0.20	3.14	3.11	3.11	3.11	3.08
		pre-phase rate 0.25	3.11	3.11	3.11	3.15	3.11
		pre-phase rate 0.30	3.14	3.09	3.09	3.09	3.09
		pre-phase rate 0.35	3.1	3.09	3.15	3.2	3.11
		pre-phase rate 0.40	3.07	3.1	3.12	3.07	3.11
	phase rate 0.15	pre-phase rate 0.05	3.13	3.11	3.08	3.15	3.12
		pre-phase rate 0.1	3.11	3.12	3.11	3.15	3.11
		pre-phase rate 0.15	3.14	3.11	3.12	3.12	3.08
		pre-phase rate 0.20	3.1	3.08	3.09	3.1	3.13
		pre-phase rate 0.25	3.11	3.12	3.12	3.15	3.12
		pre-phase rate 0.30	3.1	3.1	3.12	3.16	3.15
		pre-phase rate 0.35	3.08	3.09	3.15	3.09	3.12
		pre-phase rate 0.40	3.1	3.12	3.13	3.14	3.12
	phase rate 0.20	pre-phase rate 0.05	3.08	3.1	3.1	3.11	3.12
		pre-phase rate 0.1	3.1	3.09	3.14	3.11	3.13
		pre-phase rate 0.15	3.11	3.08	3.17	3.09	3.18
		pre-phase rate 0.20	3.11	3.12	3.11	3.14	3.13
		pre-phase rate 0.25	3.15	3.12	3.14	3.1	3.11
		pre-phase rate 0.30	3.18	3.14	3.28	3.21	3.21

	pre-phase rate 0.35	3.25	3.22	3.24	3.24	3.21
	pre-phase rate 0.40	3.23	3.23	3.23	3.52	3.43

Table 6: Pipeline 1 Run Times

Table 7 shows the results obtained from the run times from pipeline 2. As can be seen in the table the run times are fairly consistent through cycles, but show some variation through different values of phasing and pre-phasing. The run times show no real correlation and not much variations ranging from 42.06 to 44.46, meaning that the run time is not dependent on the error seen in the data.

Simulation Table for Pipeline 2: Run Times							
			Run 1 (sec)	Run 2 (sec)	Run 3 (sec)	Run 4 (sec)	Run 5 (sec)
			cycles 3_12	cycles 1_20	cycles 20_50	cycles 3_33	cycles 50_70
clean	clean	clean	43.04	44.36	43.96	44.12	43.44
decay rate 0.02	phase rate 0.05	pre-phase rate 0.05	43.04	44.36	43.96	44.12	43.44
		pre-phase rate 0.1	44.15	43.9	43.53	43.49	43.51
		pre-phase rate 0.15	43.34	43.91	43.41	43.47	43.65
		pre-phase rate 0.20	43.67	43.35	43.27	43.43	43.17
		pre-phase rate 0.25	43.4	43.44	43.28	43.31	43.13
		pre-phase rate 0.30	43.81	43.26	43.31	43.41	43.2
		pre-phase rate 0.35	43.25	43.54	43.35	43.28	43.2
		pre-phase rate 0.40	43.2	43.25	43.27	43.24	43.34
	phase rate 0.1	pre-phase rate 0.05	43.06	43.19	43.2	43.34	43.32
		pre-phase rate 0.1	43.32	43.47	43.38	43.58	43.48
		pre-phase rate 0.15	44.41	43.3	43.45	43.33	43.46
		pre-phase rate 0.20	43.23	43.49	43.42	43.44	43.19
		pre-phase rate 0.25	43.27	44.09	43.46	43.47	43.36

		pre-phase rate 0.30	43.17	43.28	43.29	43.25	43.45
		pre-phase rate 0.35	43.36	43.42	43.12	43.57	43.4
		pre-phase rate 0.40	43.4	43.38	43.4	43.28	43.33
	phase rate 0.15	pre-phase rate 0.05	43.63	43.31	43.44	43.57	43.37
		pre-phase rate 0.1	43.28	43.69	43.47	43.47	43.34
		pre-phase rate 0.15	43.15	43.29	43.24	43.4	43.05
		pre-phase rate 0.20	42.97	43.05	43.42	43.59	43.57
		pre-phase rate 0.25	43.57	43.7	43.37	43.28	43.41
		pre-phase rate 0.30	43.3	43.53	43.47	43.53	43.47
		pre-phase rate 0.35	43.25	43.24	43.37	43.4	43.48
		pre-phase rate 0.40	43.56	43.47	43.62	43.43	43.42
	phase rate 0.20	pre-phase rate 0.05	43.37	43.42	43.45	43.21	43.42
		pre-phase rate 0.1	43.35	43.39	43.22	43.58	43.45
		pre-phase rate 0.15	43.38	43.12	43.26	43.24	43.32
		pre-phase rate 0.20	43.33	43.32	43.24	43.42	43.49
		pre-phase rate 0.25	43.44	43.2	43.29	43.61	43.24
		pre-phase rate 0.30	43.21	43.52	43.64	43.27	43.53
		pre-phase rate 0.35	43.25	43.29	43.39	43.25	43.38
		pre-phase rate 0.40	43.93	43.91	43.54	43.88	43.84

Table 7: Pipeline 2 Run Times

After obtaining run times for both pipeline 1 and 2 a graphical representation of the results were created, as seen in figure 10. As can be seen no clear pattern in either graph can be discerned. Both seem to have values which are relatively consistent with one another as phasing and pre-phasing increases. Although the distribution of each graph is fairly constant, the average of each pipeline is quite different. As can be seen in the graphs below, pipeline 1

has run times ranging from 3.05 seconds to 3.46 seconds. This is relatively fast when compared to the run time for pipeline 2 which shows ranges from 42.8 seconds to 44.6 seconds. This is about 10x more than pipeline one. Thus from the data pipeline 2 seems to have a much slower run time then pipeline 1.

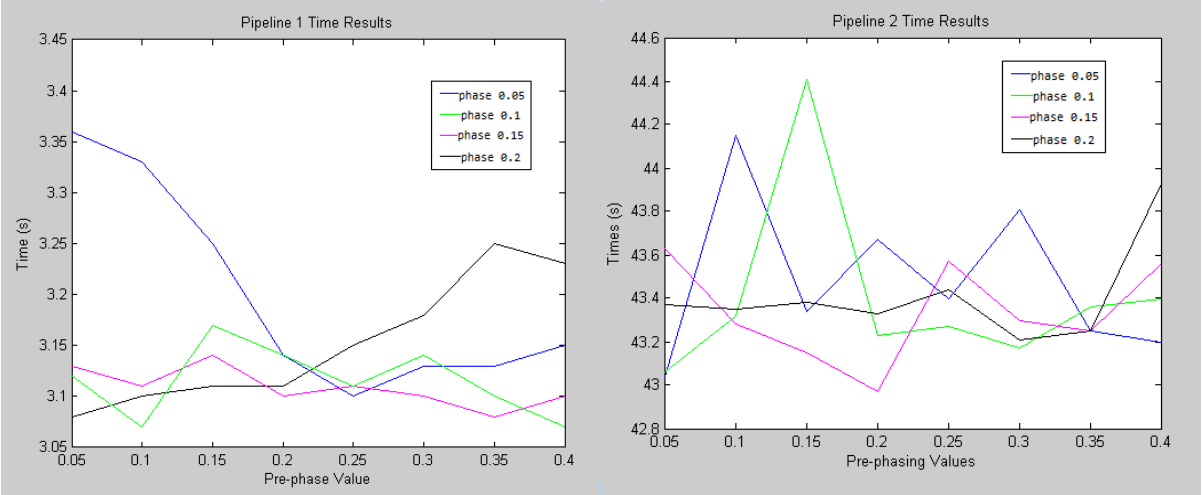

Figure 10: Comparison of Timing for the Pipeline 1 and 2

5.3 Autoregressive phasing results

The last was created using an Autoregressive function to generate phasing and pre-phasing values. Table 8 below shows the result of this method, the phasing values simulated into the data are seen in column 1 and the Pre-phasing simulated vales are seen in row 1. These results show that the autoregressive method is able to incorporate both the phasing and pre-phasing values. This means that this method will show similar results to that seen in pipeline 2, but at a much slower rate taking about 1 hour to acquire each value seen in the table below.

Autoregressive Resultants									
Phasing Values	***Pre-phasing Values***								
	0	0.05	0.15	0.25	0.35	0.1	0.2	0.3	0.4
	0.05	0.09359	0.226005	0.368907	0.489537	0.155924	0.297441	0.431909	0.529938
	0.15	0.225987	0.383319	0.536544	0.636073	0.301808	0.461786	0.596169	0.646961
	0.25	0.369643	0.537284	0.670418	0.701535	0.45274	0.611401	0.701503	0.66094
	0.35	0.492034	0.635109	0.698417	0.624131	0.567568	0.681698	0.679815	0.534985
	0.1	0.089827	0.221997	0.36494	0.485611	0.151798	0.293363	0.427396	0.52411
	0.2	0.221439	0.378778	0.531639	0.631077	0.29724	0.457095	0.591183	0.64165
	0.3	0.364757	0.532045	0.665773	0.697202	0.447455	0.606448	0.696882	0.656921
	0.4	0.487375	0.630702	0.694419	0.621753	0.562661	0.677186	0.676743	0.533009

Table 8: Autoregressive Resultants

CHAPTER 6-DISCUSSION

6.1 Data Simulation

6.1.1 Sequence simulation

One of the most important parts of this DNA sequencing project was data simulation. There are readily available sequence intensity data files online but the original sequence is not available. Therefore, the team developed a sequence simulator from which the intensity data files will be further simulated. The first attempt to simulate a DNA sequence was by using excel's random function. In each cell, a random function was placed on A,G,C,T while the number of cells are equal to the length of the sequence, the output csv/xlsx file being a sheet of cells containing nucleotides. This attempt seemed to work at first but it was found that whenever the excel file is opened, the cells get randomized again. Also, the team's data comparator(base call vs real sequence) had issues interpreting the csv/xlsx files. The team found another solution which was to simulate a txt file directly from python. Same approach was used to randomize the nucleotides across the sequence and the final output file is stored as a txt file. The nucleotides simulated in the txt files do not change anymore as the txt files are permanently simulated from python. The data comparator was also able to directly read the data from txt files.

6.1.2 .Cif simulation(clean)

After having successfully simulated dna sequence .txt files, .cif files were simulated from the .txt files. Illumina stores intensity data in files with .cif extension, where the intensities of clusters are stored in bytes as little endian 2 byte integers. The team followed the same data storage format and simulated similar .cif files. The team simulated clean data without any phasing or noise first by setting the highest base callable intensity significantly higher than the other three intensities for a single cluster. The team set the intensity ranges in a manner that

the overall chastity for the intensities of each cluster is 0.7 so that those intensities pass Illumina's chastity filter at 0.6. The team ran the clean .cif files through the base caller and afterwards, through the comparator to compare with the original sequence .txt files. It was found that there were zero missed base calls which confirmed that the data was clean.

6.1.3 Cif simulation(decay)

The team assumed that there would be source(s) of noise interfering with the cluster intensities other than phasing. One possible source of noise, decay, was assumed since the fluorescent dyes attached to the nucleotides can decay with time with exponential differentiation $y = a(1-b)^x$ where y is the intensity at x cycle, a is the intensity of the first cycle, b is the decay rate and x is the number of current cycle.

Different color dyes are used for different nucleotides and thus, there were assumed to be four different decay rates. Even though this decay nature changes the intensities of the clusters, all four intensities in a cluster are reduced so there are no obviously notable interference with the base calls. Intensity data with different decay rates were tested, base-called and compared with the original sequence simulated. The large decay rates introduced a small error rate, and the small decay rates did not have an effect on the final base call. However, combined with phasing noise, the decay noise had a much more notable interference with the base calls.

6.1.4 Cif simulation(phase)

The main reason why base call results are inaccurate these days is the noise that comes from phasing.

Phasing is the lagging of a fixed fraction of molecules in each cluster at each cycle, in the sense that those molecules fall one base behind in sequencing. Pre phasing is essentially the same process, the only difference being that the molecules fall one base ahead.

In simulating the phased .cif files, the team incorporated the phase equation provided by Illumina,

$\binom{n}{k}(1-p)^{n-k}p^k$, to the intensities (n is the number of current cycle, k is the number of phased cycles,p is the phased fraction). Similar equation was incorporated for prephasing in an opposite direction to the intensities as well.

The change in values of the phase fraction and prephase fraction were discovered to have a huge impact on the accuracy of the intensities. The larger the phase fraction/prephase fraction, the more errors were introduced in the base calls.

6.1.5 Direct hdf5 simulation

The team's Matlab script that converts .cif to hdf5 format, was found to be overly time consuming in verifying with more than a thousand sets of data. Therefore, the quicker direct hdf5 simulation with desired characteristics was introduced. Python has a package, h5py, that can store simulated data in h5 format. Thus, the team used the package to simulate all the clean and noise data again that was previously simulated as .cif. Those h5 files were base-called directly and compared with base-called results of hdf5 simulated from cif, and it was confirmed that they both contained the same intensities.

6.2 Alternative Data Sources

Beyond simulated data, data was also acquired from two alternate sources. One data set was attained directly from an Illumina sequencing run, while another set containing a phi X 174 control was downloaded from an online server. Ultimately, the post-conceptual revisions of each pipeline were tested and evaluated using the simulated data. Some problems were encountered that made attempting to analyze the sequencer data difficult. The tags used to identify the known phi x control contained within the sequence were not available and time was not allocated to search for the sequence. The server data was not used for pipeline testing purposes because it was found to be of very poor quality. Early on, the data was called and aligned to the known phi x sequence and very few segments had a good alignment ratio. This indicates that the error rates are large. Unfortunately, the source of this error is unknown and could impact the validity of the pipeline. It was then decided that developing a simulation script to produce controllable data sets with known error rates would be the most effective way of testing and validating the pipeline.

6.3 .Cif Conversion to HDF5

In order to use the data that was simulated or received from Stanford, the team needed to create a converter so that the data would be in the proper format. This was done using matlab and then python. .Cif are stored in binary format so the converter changed those into integer values and then converted them to HDF5. HDF5 is a file format that is used to store very large amounts of data in different “files” inside one shell file. This was very useful as the illumina machines output numerous .cif files per read.

6.4 Phasing and Pre-Phasing

The phasing and pre-phasing process of the pipeline was modified three different times. The first pipeline created uses the tech documents provided from the Illumina website. This document had lots of unknowns including the noise error and phase error calculations. Then pipeline 2 was created using known phasing and pre-phasing values from the simulation data. Then for pipeline 3 an autoregressive model was created and run through. The results of these changes will be discussed below.

6.4.1 Results from Pipeline 1

For pipeline 1 the illumine tech document was replicated. Two main components of the pipeline were not specifically discussed. The first being the noise error variable, this variable takes into account the fading that occurs towards the last cycles of the run, also known as bleaching. This component was determined to be modeled on an exponential decay curve. The next unknown variable from the tech document was the error due to phasing. Neither the tech document nor the Illumina web page provided any information on how the phasing error variable was calculated. This lack of information meant that the phasing variable had to be estimated using a random variable.

Since the Illumina pipeline is based on cascading variables from one function to another, the results from pipeline 1 resulted in a random number generator. Thus the percentage of error was randomly high in one set of data and then low in another set of data, this is seen in figure 9. The graph shows no discernible pattern when comparing the lowest phasing and pre-phasing values as opposed to the highest phasing and pre-phasing values. Thus this data was labeled as randomized and not analyzed further for higher or lower percentage errors were done.

The run time for pipeline 1 did show fast run time values, as seen in figure 10. The run time on average is about 3 seconds for all ranges of phase and pre-phasing values. This is a positive to this type of analysis meaning that it will provide rapid results and will be able to handle data with higher errors just as efficiently as data with minimal error. Thus with pipeline 1 although it does produce high percentage errors, the processing time is relatively quick.

6.4.2 Results from Pipeline 2

For pipeline 2 the phasing and pre-phasing ratio of the data was taken from the simulated data and imported into a polynomial summation process. This process involved imputing known quantities of phasing and pre-phasing variables and then placing these values into a coefficient for a polynomial expression based on the cycle being observed. This process involved iterating through the data both forward for phasing and backwards for pre-phasing.

The results of this pipeline was is seen in figure 9 As can be seen this pipeline shows a relatively linear pattern being that the higher phasing and pre-phasing values result in the higher percent error results and the lowest resulting in the lowest percent error. This creates a positive relationship between the phasing and pre-phasing values and the percent error values. This data correlates with the idea that higher simulated error will result in higher percentage error meaning in comparison to pipeline 1 this result is much more realistic and a lot more efficient in terms of percent error. The highest value for pipeline 2 being 15% isn't even the lowest possible value for pipeline 1. When comparing pipeline 1 and pipeline 2 percent error results the clearly more efficient method is seen in pipeline 2.

In terms of run time the results can be seen in Figure 10. As can be seen the pipeline shows an average run time of about 45 seconds. This is approximately about 10X slower than

that of pipeline `1. Although this is a slower pipeline as compared to pipeline 2, it is a lot more effective in terms of percent error. It was determined that pipeline 2 was the optimal pipeline because of the high performance of the percent error. In terms of the data provided it is a lot more helpful for a pipeline to produce better data then to run at a quicker speed and thus pipeline 2 was determined to be the better method.

6.4.3 Results from Pipeline 3

The final method which was implemented was the Autoregressive method. This involved importing individual clusters and individual channels through all cycles into an autoregressive function. The average coefficient value for the autoregressive output was calculated for each data set. This resulted in a value corresponding to the sum of the phasing and pre-phasing values input into the simulated data.

The result of this pipeline can be seen in table 8. The phase values can be seen in the first column and the pre-phase values can be seen in the first row of the chart. These values correspond to the amount of phasing and pre-phasing generated by the simulated data. As can be seen the autoregressive method is fairly accurate. The sum of the input phasing and pre-phasing values correspond fairly closely with the calculated autoregressive output coefficient. This means that the results of this method will show a similar trend to that seen in pipeline 2.

6.5 Basecalling

One of the team's initial goals was to develop a method for quickly and efficiently producing the nucleotide sequence of a given intensity set. This was considered a vital component of the final pipeline because validation of the pipeline's effect on intensity data was to be based off the direct decrease in erroneously called bases in the resulting sequence. The initial basecalling

script was designed to take a .CIF file as an input. A .CIF file is a proprietary file produced by Illumina Sequencers, which stores the intensity values for particular channels in a binary format. The prototype basecaller converted these .CIFs from binary and output a .txt file containing each cluster on its own line. When the scope of the project shifted to focus on generated data, the basecalling script was rewritten to use a three dimensional array of intensities as the input. By using a standardized 3D array, the team could format both raw generated data and corrected data to be compatible with the basecaller.

6.5.1 Functional Techniques Utilized

The most crucial component when producing a sequence from intensity values is the structure that the data is stored in. The team determined that the most efficient method for storing the massive amounts of data produced by the sequencers would be in three dimensional arrays. Once the data was formatted in three dimensions, separate functions in the pipeline would be able to analyze sections, or "slices" of the array. Using this slicing method, the basecaller is able to read in all four channels of a cluster at a particular cycle. The maximum of these four intensities is the most prevalent nucleotide for that cluster at that cycle. The index of each channel correlates to the channel's nucleotide (A, C, G, T). The index of the maximum is appended to a text file and once all the channels in a cluster have been analyzed, a newline is created within the text file.

6.5.2 Output Files

While the original intent of producing an accurate basecalling mechanism was achieved, there is still more that can be done in terms of data readability. Presently, the data is stored in a text file, which is acceptable for the team's purposes, but if the sequence analysis is to be taken

further then a more universal format would need to be used. The team investigated the implementation of a FASTA format to increase the cross-program utility of the output files, but never proceeded to implement the format into the pipeline. The information included in the FASTA format is not necessarily information that is useful when doing pure error analysis of the sequence clusters. FASTA format is able to indicate uncertainty when representing nucleotides in sequence. For example, a specific character can indicate positions which only have potential to contain a base with amino groups (A & C) while another character indicates a base that only has the potential to be a ketone (G & T). The team's pipeline does not implement any technique for limiting possible bases in any given position so it was determined that there would be very little present advantage to applying the FASTA format. In the future it may be worth investigating the implementation of this or other formats to allow the output file to be run through other analysis or alignment programs.

6.6 Data Confirmation

It was determined that the most effective metric for determining the efficacy of the pipeline would be the reduction in error rate. To measure the error rate of a sequence, the data confirmation program was developed. This program derives a percentage from the amount of errors seen over the total number of nucleotides in a sequence. Each sequence was compared to the origin sequence, which can be assumed to have zero errors. The simulated sequence will produce a specific amount of error, while the post-pipeline sequence should have a lower error rate. The data confirmation program is also capable of providing the index of the errors to help identify trends in the data.

CHAPTER 7- FINAL DESIGN AND VALIDATION

7.1 Final Design

This project resulted in three pipelines, which each dealt with sequence errors produced by phasing in next generation sequencers. All three pipelines were based around the same sequence of operations with a varying phase correction component. The first module of each pipeline was the simulator, used to simulate data sets with varying degrees of error. The next process was the respective phase correction method. The final two steps were the base calling program and the data confirmation program, which provides an error rate for the sequence.

7.2 Validation of Phase Correction

In order to validate each method of phase correction implemented in the pipelines, a metric to appraise the reduction in error was necessary. It was determined that the most effective process for rating the error reduction in the analyzed sequence would be a direct comparison with the initial, uncorrected sequence. A script was developed to compare the sequence txt. file, containing either the simulated or post-pipeline sequence, with the sequence txt. file used to simulate the intensity data. Equations for calculating the error rates are found below.

$$\frac{Miscalls\ in\ Simulated\ Sequence}{Total\ Number\ of\ Bases\ in\ Sequence} * 100 = Simulated\ Error\ Rate$$

$$\frac{Miscalls\ in\ Post{-}pipeline\ Sequence}{Total\ Number\ of\ Bases\ in\ Sequence} * 100 = Pipeline\ Error\ Rate$$

$$Simulated\ Error\ Rate\ - Pipeline\ Error\ Rate = \boldsymbol{Reduction\ in\ Error\ Rate}$$

The simulated error rate is calculated by dividing the number of differences between the simulated sequence and the origin sequence by the total number of nucleotides in the sequence. This value describes the error content of the simulated data set, which can be controlled by variables within the simulator program. The pipeline error rate is calculated by dividing the number of differences between the post-pipeline sequence and the origin sequence by the total number of nucleotides in the sequence. This value describes the error content of the sequence after it has been adjusted by the pipeline. In theory, if the pipeline is valid this value will be less than the simulation error rate. The difference between the simulated error rate and the pipeline error rate is the reduction in error rate, resulting from the pipeline.

7.3 Validating the integrity of simulated data

As mentioned above in Chapter 4, the team simulated direct hdf5 files instead of .cif files as the conversion process takes a huge amount of time. The hdf5 files generated were run through the base caller and sequence comparator to check whether the sequence data completely matches the data contained in the .cif files they were generated from. The simulated data was validated as the h5 files and .cif files were found to contain the exact same data.

Also, the team validated the error simulation by basecalling the h5 files containing phase/prephase errors, and decay errors. It was confirmed that the clean data also got zero error percentage in base calling. It was discovered that the data with decay error alone did not have a high error percentage. Only large decay rates induced a small error percentage in the data. This validates the simulated data as the decay nature alone is not supposed to affect the overall intensity of a cluster as all four channel intensities of that cluster will be reduced without making a notable difference between each of the individual channel intensities. However, the data with phase/prephase error alone seemed to have a bigger impact on the base call accuracy giving larger error percentages as the phase/prephase parameter values get larger. This also validates the simulation of data with phase/prephase error since the phasing /prephasing nature, where the intensity changes between all four channels at one cycle are dependent on the intensities of previous/next cycles, is likely to affect the data accuracy more than decay, where the intensities are changing at a uniform rate. For the data with both decay and phase/prephase error combined, the error percentages were slightly larger, which also apparently validates the simulation as the combination of the two noises should give a larger noise.

CHAPTER 8 – CCONCLUSIONS AND RECOMMENDATIONS

Pipeline 2 depicted the best results in terms of accuracy. Although pipeline 1 shows a faster run time, the high percent error makes it a very ineffective pipeline and would not be recommended for any type of sequencing data. Pipeline 2 on the other hand may take longer to run but It will supply the lab technician with a more accurate called sample. In terms a test for a genetic mutation, the curtail component is going to be acquiring accurate data. A physician and patient would sacrifice time for more accurate results especially if that mutation could result in a life or death diagnosis. Thus pipeline 2 is the most optimal pipeline. Pipeline 3 does show promise in possibly supplying quality data which is comparable to pipeline 2, but future testing will need to be done in order to verify this prediction.

It is this team recommendation that future work be completed on the autoregressive function as well as the functional runtime of the pipeline. This process shows great promise but requires more work.

Works Cited

Aksyonov, S., M. Bittner, L. Bloom, L. Rehakrantz, I. Gould, M. Hayes, U. Kiernan, E. Niederkofler, V. Pizziconi, and R. Rivera. "Multiplexed DNA Sequencing-by-synthesis." *Analytical Biochemistry* 348.1 (2006): 127-38. *ScienceDirect.cm*. 1 Jan. 2006. Web. 9 Oct. 2013

Ewing, B. & Green, P. Base-calling of automated sequencer traces using phred. II. Error probabilities. Genome Res. 8, 186-194 (1998).

Ewing, B., Hillier, L., Wendl, M.C. & Green, P. Base-calling of automated sequencer traces using phred. I. Accuracy assessment. Genome Res. 8, 175-185 (1998).

Illumina. "An Introduction to Next Generation Sequencing Technology." N.p., n.d. Web. 11 Oct. 2013.

Illumina. "Genetic Epidemiology with MiSeq: Tracking Influenza H7N9 in China ." *Public Health Research Update* (2013).

Goldenberg, Oliver. "Illumina Sequencing Worlkflow HiScanSQ." *Illumina*. N.p., 2010. Web. 11 Oct. 2013. <http://lhmg.amu.edu.pl/text/sequencing/Illumina%20Sequencing%20Workflow%20HiScanSQ.pdf>.

Ji, Hanlee, and Jay Shendure. "Next-generation DNA sequencing." *Nature Biotechnology* 26.10 (2008): 1135+. *Academic OneFile*. Web. 9 Oct. 2013.

Kao WC, Stevens K, Song YS: BayesCall: a model-based basecalling algorithm for high-throughput short-read sequencing. Genome Research 2009, 19:1884-1895.

Kolata, Gina. *DNA Test for Rare Disorder Becomes More Routine* . Health . New York : New York Times , 2013 . Newspaper .

Macevicz, S.C. DNA sequencing by parallel oligonucleotide extensions. US patent 5750341 (1998).

"Macrogen, Inc. Increases Investment in Illumina's Next-Generation Sequencing Technology." *LexisNexis Acedemic* (2012): 185 .

Massingham and Goldman: All Your Base: a fast and accurate probabilistic approach to base calling. Genome Biology 2012, 13: R13.

Minoche, A. E., Dohm, J. C., & Himmelbauer, H. (2011). Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and genome analyzer systems. *Genome Biol*, *12*(11), R112.

Morozova, O., & Marra, M. A. (2008). Applications of next-generation sequencing technologies in functional genomics. *Genomics*, *92*(5), 255-264.

Nakamura, K., Oshima, T., Morimoto, T., Ikeda, S., Yoshikawa, H., Shiwa, Y., ... & Kanaya, S. (2011). Sequence-specific error profile of Illumina sequencers.*Nucleic acids research*, *39*(13), e90-e90.

"Oncology Overview." *Oncology Overview*. Illumina, n.d. Web. 8 Oct. 2013.

Renaud, Gabriel, et al. "freeIbis: an efficient basecaller with calibrated quality scores for." *Bioinformtics* (2013): 1208-1209.

Pareek, C. S., Smoczynski, R., & Tretyn, A. (2011). Sequencing technologies and genome sequencing. *Journal of applied genetics*, *52*(4), 413-435.

"RTA Theory of Operations." *Illumina*. Illumina, Inc., 9 May 2011. Web. 5 Oct. 2013. <http://res.illumina.com/documents/products/technotes/technote_rta_theory_operations.pdf>.

Ruan, J., Jiang, L., Chong, Z., Gong, Q., Li, H., Li, C., ... & Wu, C. I. (2013). Pseudo-Sanger sequencing: massively parallel production of long and near error-free reads using NGS technology. *BMC genomics*, *14*(1), 711.

Sinville, R. and Soper, S. A. (2007), High resolution DNA separations using microchip electrophoresis. J. Sep. Science, 30: 1714–1728. doi: 10.1002/jssc.200700150

"Solexa Technology." *Solexa Technology*. N.p., n.d. Web. 11 Oct. 2013.