An Investigation into the Analysis of TOF-SIMS Data

A Thesis Submitted in Partial Fulfillment of the Requirement
for the Degree of Bachelor of Science with Honors in
Physics From the College of William and Mary in Virginia

by

Brian Christopher Spencer

Accepted For _____
(Honors, High Honors, Highest Honors)

_____
Advisor: Dr. William Cooke

_____
Dr. John Delos

_____
Dr. Jan Chaloupka

_____
Dr. Michael Trosset

Abstract

Over the course of the past year, investigations were made into the possibility of resolution improvement for the time of flight data recorded by a TOF-SIMS (Time of Flight Secondary Ion Mass Spectroscopy) apparatus. Several attempts were made to detect and remove inhomogeneous broadening caused by time delays of the detection of ions. Initially, global shifts were detected between two regions of an image and removed from the mass spectrum of various samples. More advanced regionally specific investigations were made on a sample of two Rtil (a conductive solution) droplets in close proximity to characterize and remove the unusual broadening of the F and PF_6 peaks present in the sample spectrum. Using these techniques resolution improvement was seen and the resolution of individual peaks was improved by up to a factor of five.

Acknowledgements

I would like to thank Dr. Haijian Chen for his peak picking program and the helpful information he gave throughout the course of my research. Furthermore, I wish to thank Pete Harris and the rest of the MassSpec group for their advice throughout the year. Additionally, I wish to thank Dr. John Delos, Dr. Jan Chaloupka, and Dr. Michael Trosset, for acting as my review committee and providing valuable insight into my project. Finally, I must thank Dr. William Cooke for his endless advice, invaluable information, and constant supply of motivation to make the most of this project.

Contents

Background

The ability to determine the presence and concentration of various chemicals on a biological sample is a major goal of the scientific community. Time-of-Flight Secondary Ion Mass Spectrometry (TOF-SIMS) is one method through which this might be achieved. When utilized to its full capacity, this technique could offer new solutions to problems in many fields including medical diagnostics, biotechnology, and molecular electronics. However, this technique needs further refinement before these applications can become a reality [Winograd, 2003].

The TOF-SIMS process uses a primary ion beam (in our case either gold or gallium) to eject secondary ions from a sample. These secondary ions are then accelerated through an electric potential towards a detector. The detector records the time of flight of each ion which hits its surface, along with the position of the ion beam that produced the event. Ideally, from this information we can calculate the mass of the ion and its original position on the sample.

Unfortunately, there are several factors, in particular sample surface variations, which lead to deviations in the TOF-SIMS data. Ideally, all ions with the same mass would travel through the potential at the same speed, arriving in the same amount of time. Realistically, the variations in the sample, ranging from charge buildup to surface height fluctuations, cause a spread in arrival times. When these times are converted to mass, we obtain a mass range, as opposed to a specific mass, for a group of identical ions. Depending on the quality of the sample, this decrease in mass resolution can be significant, creating difficulty in ion identification [Aubriet, 2003]. This problem is compounded when dealing with large molecules because they tend to break apart during the extraction phase of the TOF-SIMS process.

Therefore, a technique for detecting these variations, and accounting for them, is crucial for the utilization of the TOF-SIMS instrument.

Introduction

Several attempts have already been made to develop methods for improving mass resolution. One such technique has been to place a mass matrix onto the substrate then lay the chemical sample onto the known mass matrix. The use of a mass matrix, typically consisting of large crystals, has been shown to help prevent the fragmentation of large molecules, but at the cost of decreased resolution. Attempts have been made to characterize the topography of a sample using the shifts from the known mass matrix ions and align the other sample ion accordingly, improving resolution [McDonnell, 2003].

My method of alignment identifies and removes the time shifts of the detections using ions already present in the sample, removing the need of the mass matrix in the alignment process. The hope is that the large molecules which break apart will form similar fragments every time they do break down. Then, by looking for these fragments we can detect the presence of the larger molecule in the original sample. By removing the need of the mass matrix we remove the inherent resolution decrease it causes.

The techniques which I have applied to improve mass resolution can be broken into two main methodologies. My first attempt was to identify the presence and position of a large number of peaks in a sample, detect the shifts between these peaks from different regions, and account for these shifts to remove inhomogeneous broadening. Inhomogeneous broadening results when individual spectra from various regions, which have been shifted in time and are thus unaligned, are combined to create a total spectrum. While the total number of ion counts

remains constant, the combination results in a wider peak because it consists of a group of smaller unaligned peaks.

To employ this method, I first had to be able to identify the peaks and obtain their positions. To this effect, I utilized a peak picking program written by Dr. Haijian Chen (as part of his graduate work at the College of William and Mary), along with several programs I wrote to help the peak picker successfully identify a large variety of peaks (both in shape and size). Once the peaks were identified, the shifts could be detected, and removed. By adding together the aligned smaller peaks, resolution is improved.

My second method attempted to use just one ion's peak to characterize the spatially dependent shifts found in a more complicated sample. Once these shifts were understood, I could use the information to align the detections corresponding to that ion and improve its peak resolution. Furthermore, by determining how different ions are affected by the spatially dependent shifts, we can take a scaled version of the known ion's shifts and align the other ion peaks as well.

Results

1.) Identification and Characterization of Unknown Peaks

a.) Peak Models

This first step of aligning our peaks was to first identify the peaks in our sample. The first sample I analyzed was a cesium chloride droplet on a silver coated silicon substrate. Essentially, this describes our best case scenario for reducing time shifts because the silicon substrate is extremely flat and the silver coating is highly conductive, reducing the possibility of a surface charge buildup. Figure 1a shows the positive ion spectrum of the cesium chloride

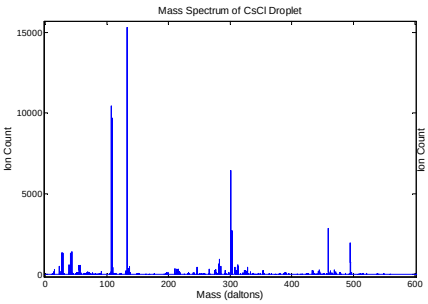

Figure 1a: CsCl Droplet Spectrum (Entire Spectrum)

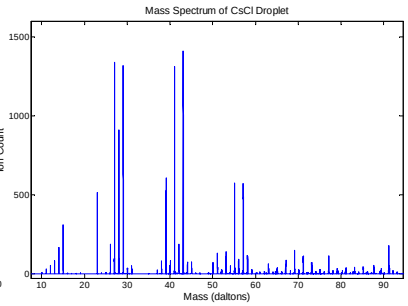

Figure 1b: CsCl Droplet Spectrum (Zoomed In)

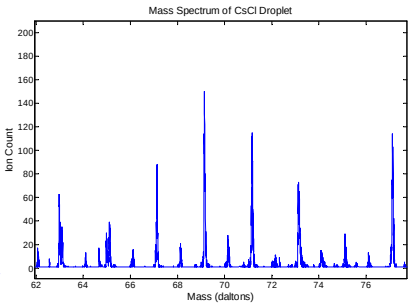

Figure 1c: CsCl Droplet Spectrum (Further Zoom In)

droplet, while figures 1b and 1c focus in on smaller peaks. Clearly we can see that there are a large number of clear peaks. Initially, it was hoped that the peak picker program of Chen would be able to successfully identify and characterize these peaks.

To identify the peaks the peak picker program utilizes the predicted line shape given in equation 1, where u is a dummy time dependent on two parameters (Resolution(R) and t_0) and the actual time (t) through equation 2.

$$\frac{1}{u^2} e^{\left(\frac{-1}{u^2}\right)} \qquad (1)$$

$$u = 1 + \frac{(t - t_0)}{(R)(1.46528)} \qquad (2)$$

Essentially, the peak picker program calculates the probability that the actual data can be described by the line shape. Since the line shape is a peak like shape, the program will not find a high probability that a flat line matches the predicted shape. However, when the actual data is shaped similarly to our peak model, the probability that it matches the predicted peak model is increased. Once the program identifies a peak, it then attempts to fit the actual data as best it can. This best fit is eventually returned by the peak picker as its approximation of the actual data near a peak. From this approximation important information like position and amplitude can be found. For example, figure 2 shows the peak pickers attempt (red) to fit the peak shown (blue).

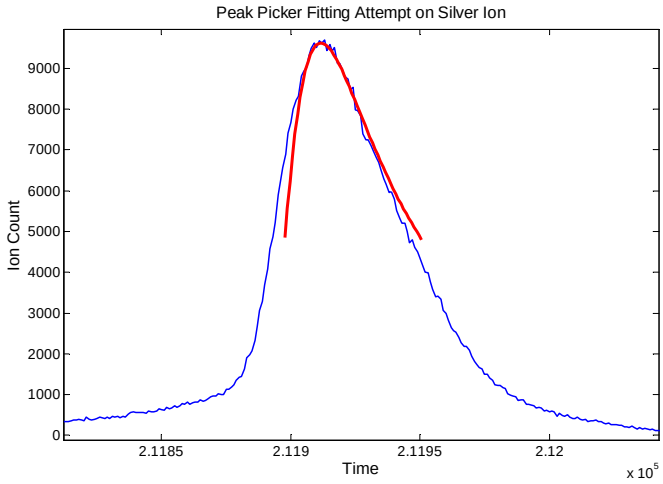

Figure 2: Peak Pickers Attempt to

b.) Optimization of Peak Picker Parameters

Our initial hope was that the original model of the peak picker would be sufficient to characterize the actual data provided by TOF-SIMS. Therefore, I attempted to develop a method of varying our two parameters (R and t_0) to best fit the calculated peaks to the actual peaks. However, it was quickly seen that while one could adjust the parameters so that a specific peak matched well, this would have a negative effect on the rest of the fits.

The next step was to characterize the error between the calculated and actual peaks, and then attempt to find the parameters which would create the largest reduction of error for the largest number of peaks. To characterize the error an equation was developed to determine the difference between the actual data and the fit provided by the peak picker (equation 3).

$$Error = \frac{\sqrt{\frac{\sum \left(y_{aj} - y_{pi}\right)^2}{N}}}{\frac{\sum y_{pi}}{N}} \qquad (3)$$

For this equation we are summing over a window around the peak, y_{ai} is the actual ion count at a given time in the window, y_{pi} is the peak picker's prediction of the ion count at that

time, and N is the total number of counts in the window. I wrote a program which could use the windows of the peaks determined by the peak picker, and calculate the error for each fit (Serror.m, Appendix A.1). However, after working with the equation for a while it was found that it could be improved. Essentially, even if our predicted model exactly described our actual data, we still would be left with the Poisson noise which is created in any counting apparatus such as TOF-SIMS. So in reality, our current measurement of error was measuring both the failure of the peak picker to approximate the curve and the noise in our data. Since the average error associated with Poisson noise is understood (it is proportional to the square root of the number counts being made), we simply calculated what we would expect it to be for each peak and removed it.

Essentially, if we assume what we are actually observing (y_{ai}) is exactly our predicted model (y_{pi}) in addition to the Poisson noise to be expected, then in our current error equation, y_{ai}-y_{pi} would simply be the Poisson noise. Since the average of the Poisson noise over the peak is equal to the square root of the number of counts (which is true for all Poisson noise), we can substitute this formula into our original error equation (eq. 3) to obtain the Poisson Error of each of our individual peaks as seen in equation 4.

$$Poisson\,Error = \frac{\sqrt{\frac{\sum\left(y_{aj}-y_{pi}\right)^2}{N}}}{\frac{\sum y_{pi}}{N}} = \frac{\sqrt{\frac{\sum(PoissonNoise_i)^2}{N}}}{\frac{\sum y_{pi}}{N}} = \frac{\sqrt{\frac{\sqrt{(\sum y_p)^2}}{N}}}{\frac{\sum y_{pi}}{N}} = \frac{1}{\sqrt{\frac{\sum y_{pi}}{N}}} \quad (4)$$

To modify our original error equation, we simply subtracted this Poisson Error from our original error equation. Now our new error equation (equation 5) truly measured the fitted curves failure to approximate the actual data, and not the noise in our data.

$$. \quad Error = \frac{\sqrt{\frac{\sum \left(y_{aj} - y_{pi}\right)^2}{N}}}{\frac{\sum y_{pi}}{N}} - \frac{1}{\sqrt{\frac{\Sigma y_p}{N}}} \tag{5}$$

The first attempt to choose the best possible parameters was to investigate the full width half maximum (the distance between the two points of a peak which are at half the maximum amplitude of the peak). By using the predicted line shape models of the peak picker I calculated the predicted dependence of the full width half maximum of our peaks on time. This dependence (equation 6) is linear and is determined by the parameters of the peak picker.

$$FWHM = (t - t_o)\left(\frac{(2.076311732 - 0.6110356927)^2}{R}\right) \tag{6}$$

By calculating the actual dependence in the data, and approximating it with a straight line, I was able to determine which parameters to use. Due to the noise in the data, calculating the full width half maximum can be difficult, since there may be several points on both sides of the peak which cross the half maximum point. As a rough estimate, I simply calculated the distance from the first time the half maximum point was crossed to the last time it was crossed within a window around each peak.

The program Sfwhm.m (Appendix A.2) calculates the full width half maximums of the peaks. It is worth noting that Sfwhm.m requires the peak picker to have already been run so that it may determine acceptable windows in which to search for the peak and calculate their full width half maximums. Running the peak picker with any parameters which identifies the peaks well will suffice (for example I typically use $t_0 = 0$ and R = 4000). Figure 3 shows the full width half maximums of the cesium droplet data as calculated by fwhm.m, and shows Matlab's attempt to best fit the data with a straight line.

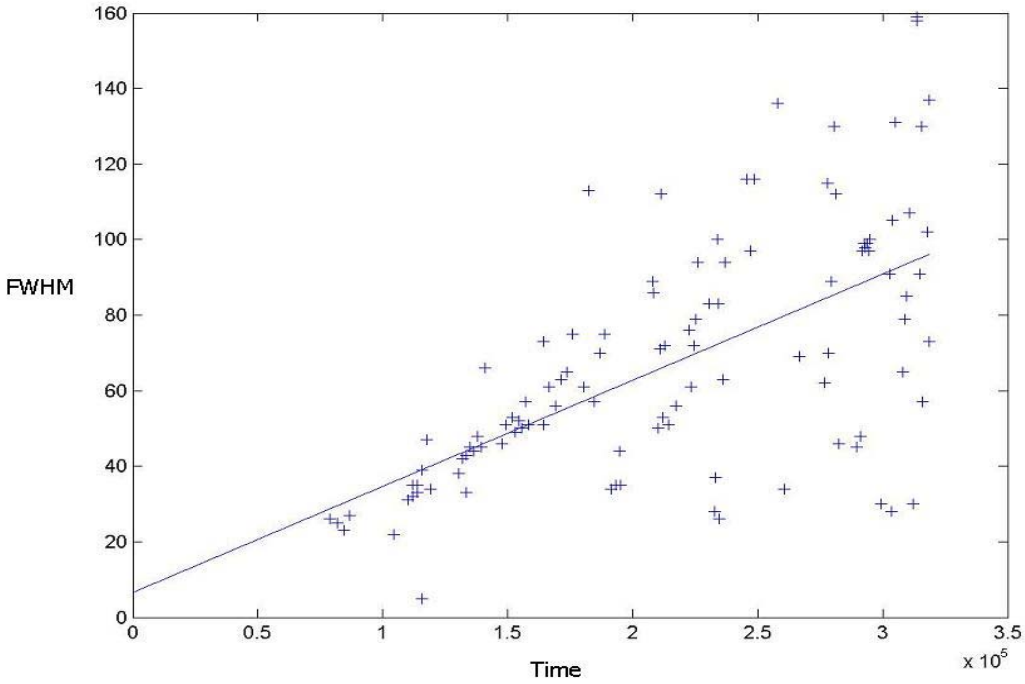

Figure 3: Full Width Half Maximums with linear best fit

Equation 7 gives the best fit to the data as provided by MatLab. From this dependence, and equation 6, we can obtain our parameter values of $t_0 = -23200$ and $R = 3558$.

$$D_{fwhm} = 0.000218t + 6.5211 \tag{7}$$

Unfortunately, these parameters did not return good approximations of our curves. In fact, the peak picker failed to identify the majority of the peaks with these parameters. This failure is most likely due to our attempt to force a linear dependence where one clearly does not exist. Attempts were made to fit just the first fifty peaks which did more closely resemble a straight line (which can be seen in figure 3). While improvement was seen, it was not enough to justify using this process to choose the parameters.

Our next attempt was to simply manually change the resolution parameter, while leaving T_0 constant at 0 with the hope of identifying some useful connection with how the resolution and errors varied. The information from this investigation is depicted on a small scale in figures 4

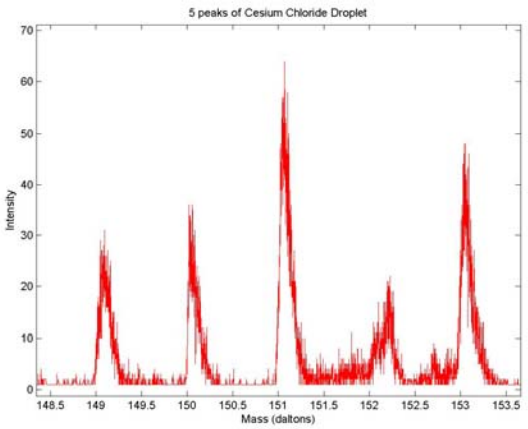

Figure 4: Five Peaks of the Cesium Chloride Droplet

and 5. Figure 4 shows five peaks in the CsCl droplet spectrum which were fitted by the peak picker. For figure 5 the peak picker was run at three different resolutions (2000, 3000, and 4000) and the errors for each of these resolutions for the five peaks was plotted. The green plus signs represent a resolution of 2000, the blue circles represent a resolution of 3000, and red stars represent a resolution of 4000. Several important features must be noted about figure 5. First, each peak does not have an error measurement for each resolution. This results when a resolution is too high to identify a peak and thus no error can be calculated. Secondly, we can see that sometimes the higher resolution produces the lowest error, and sometimes the lower resolution does. When examining all of the peaks, this trend continues. Even when two peaks are right next to each other, significantly different resolutions are sometimes needed to accurately fit them.

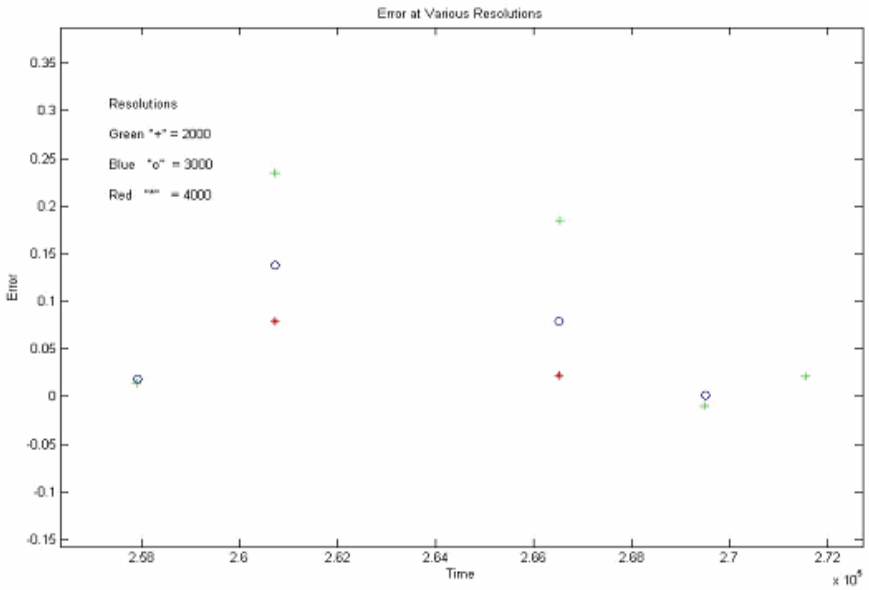

Figure 5: Errors of the 5 Peaks At Different Resolutions

At this point an entirely new approach was taken to optimize the parameters. Rather than attempting to choose one set of parameters, it was decided that we would run the peak picker at a variety of parameters and for each peak choose the resultant curve which best approximates the data. However, remembering that the purpose of the peak picker program was to reduce the data and save time, this process needed to be automated. Therefore, I wrote a program (Sbestpeaks.m, Appendix A.3) which could run the peak picker at a single resolution (the program was written so that T_0 was kept constant at 0) and compile a list of the peaks found and the errors associated with those peaks. The program then runs the peak picker for another resolution and compiles another list of peaks. Finally, a master list of peaks is created which combines the two peak lists. However, in the event that a peak appears on both lists, then only the peak with the lower error is transferred to the master list. To determine whether two peaks are in fact the same, a window is set up around the peaks in the first list, and if peaks from the second list are found within this window, then the peaks are considered to be the same. After the final list of peaks is created, the peak picker is run with another resolution, and the process can be repeated as many times as necessary for all inputted resolutions to be considered.

Figure 6 demonstrates the effect of the best peaks program. The green line represents the actual data, while the black, blue, and red lines represent the peak pickers attempt to fit the lines using a variety of resolutions (black = 2000, blue = 3000, red = 4000). The dashed line represents the fitted curve which is ultimately chosen by the bestpeaks

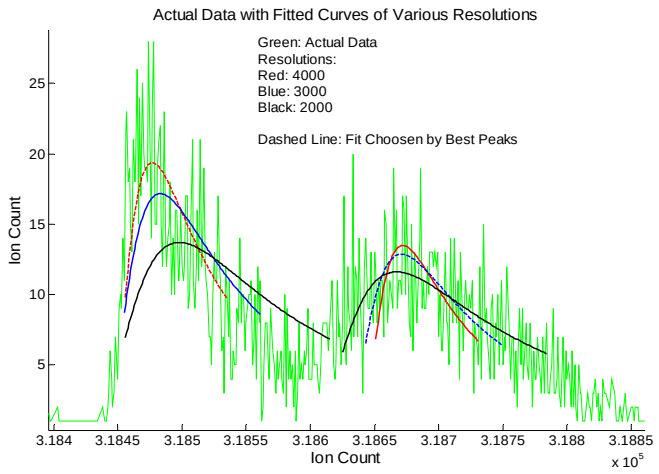

Figure 6: Application of the Beat Peaks Program

program. For the peak on the left, this happens to be the red curve, while on the right, it is the blue curve. Figure 7a and 7b demonstrates the effect this program has on our overall errors. While Figure 7a shows the errors associated with all of the different resolutions (green = 2000, blue = 3000, red = 4000, black = 6000), figure 7b shows the errors of the fitted curves which are ultimately found using the bestpeaks program. Clearly, a significant improvement can be seen.

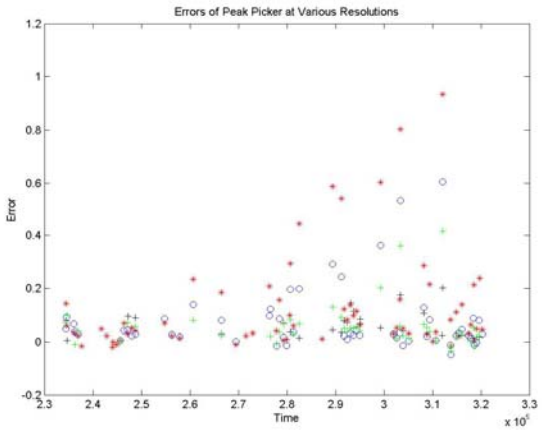

Figure 7a: Errors at Various Resolutions

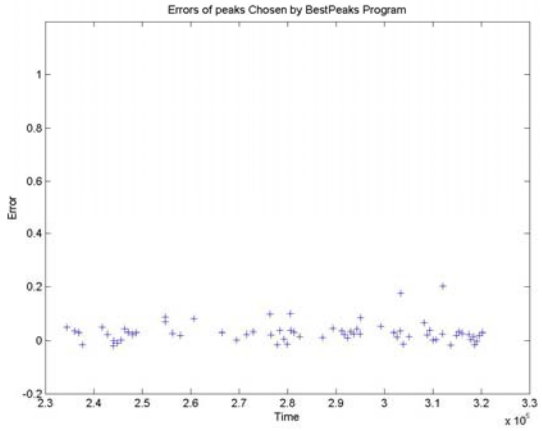

Figure 7b: Errors at BestPeaks Resolutions

c. Evaluation of BestPeaks Program

While the process of identifying our peaks was almost complete, there were still two important issues which needed to be addressed with the bestpeaks program. The first issue involved the efficient use of the program. Since the program needs to run the peak picker program for every resolution it is given, it can take a significant amount of time to run. Therefore, knowing how many resolutions to give the program to achieve the best results in the shortest amount of time is pivotal to effectively and efficiently running the program.

To evaluate this concept I first ran the peak picker individually with 6 resolutions (2000,3000,4000,5000,7000,9000) and calculated the errors associated with these resolutions. I then ran the bestpeaks program with these resolutions. By examining the peaks which required

the highest and lowest resolutions to get the best fit according the bestpeak program, I could then see whether these extra resolutions were truly necessary, or if the program would have been just as effective if they had been left out. For example, Figure 8a shows a peak which was best fit by a resolution of 9000 according to the bestpeak program. Figure 8a also shows the different attempts at various resolutions of the peak picker to fit the peak. Clearly the lower resolution fits

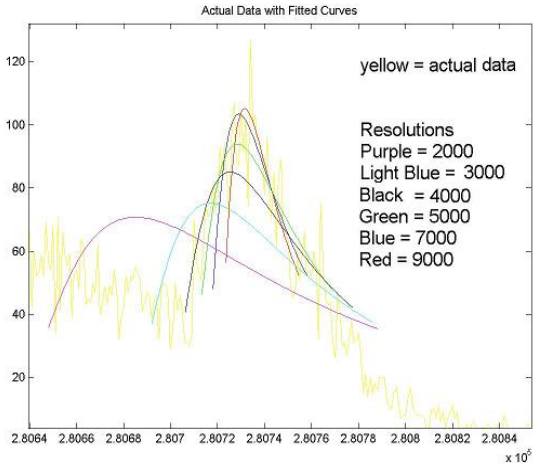

Figure 8a: Fitting Attempt of Peak Picker at High Resolution Peak

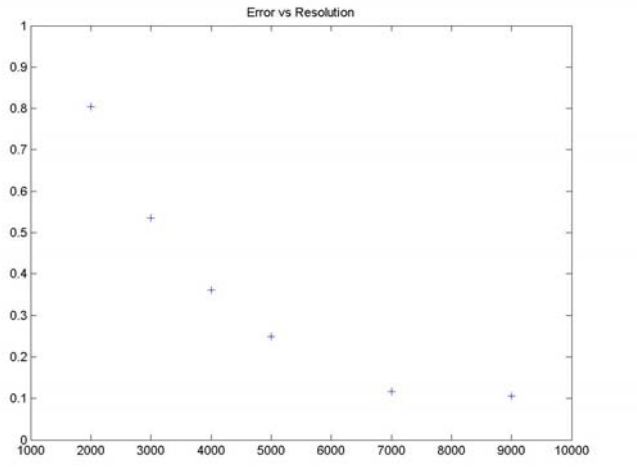

Figure 8b: Errors of Peak Picker at High Resolution Peak

are not sufficient, but we must look at the error measurements to see whether the higher resolution fits really are different in terms of their error. Figure 8b shows the error of this peak at each of the resolutions. As we can clearly see, once the resolution reaches a certain point its effect on the error saturates.

On the reverse side, figure 9a shows a peak which was best fit by a resolution of 2000, and the attempts of each resolution to fit the plot. Only two curves appear because only two resolutions actually found the peak and therefore these low resolutions were indeed necessary. However, figure 9b shows the errors associated with both of the resolutions which did find the peak. Since the error hardly changes, we can conclude that using both resolutions was not necessary.

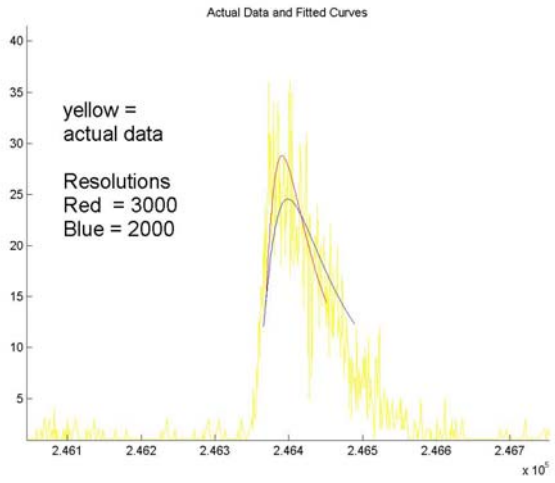

Figure 9a: Fitting Attempt of Peak Picker at Low Resolution Peak

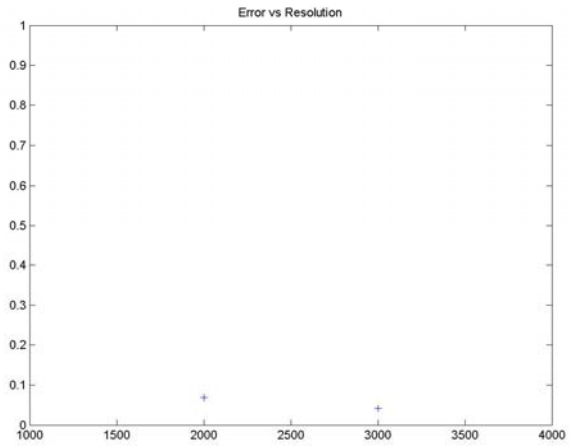

Figure 9b: Errors of Peak Picker at Low Resolution Peak

We can see from this analysis that we do need a range of resolutions, a fact we already knew. However, we can also see that using a very large amount of different resolutions is not an efficient use of time since many of the errors associated with different resolutions are very similar. The best combination appears to be a mix of a low and a high resolution, such as 3000 and 7000 with this data, which will both find the wide peaks, and provide a decent fit for both the wide and narrow peaks. In the end, which set of resolutions is right for a given set of data is dependent on that data. Thus, analyzing this trade off is an important step in utilizing the program to its full potential.

The final problem which must be addressed for the bestpeak program to work effectively is created when very high resolutions are used with the program. Higher resolutions lead to narrower fitted curves. Since the peak picker works by setting up windows around the peaks which are dependent on their width, for two peaks to be considered the same they must be closer together if their resolutions are higher (i.e the window around them is smaller). This tends to lead to problems like the one seen in figure 10. While the actual peak itself is being fit nicely by the best peak program, the program is also finding another peak inside the tail. Had the

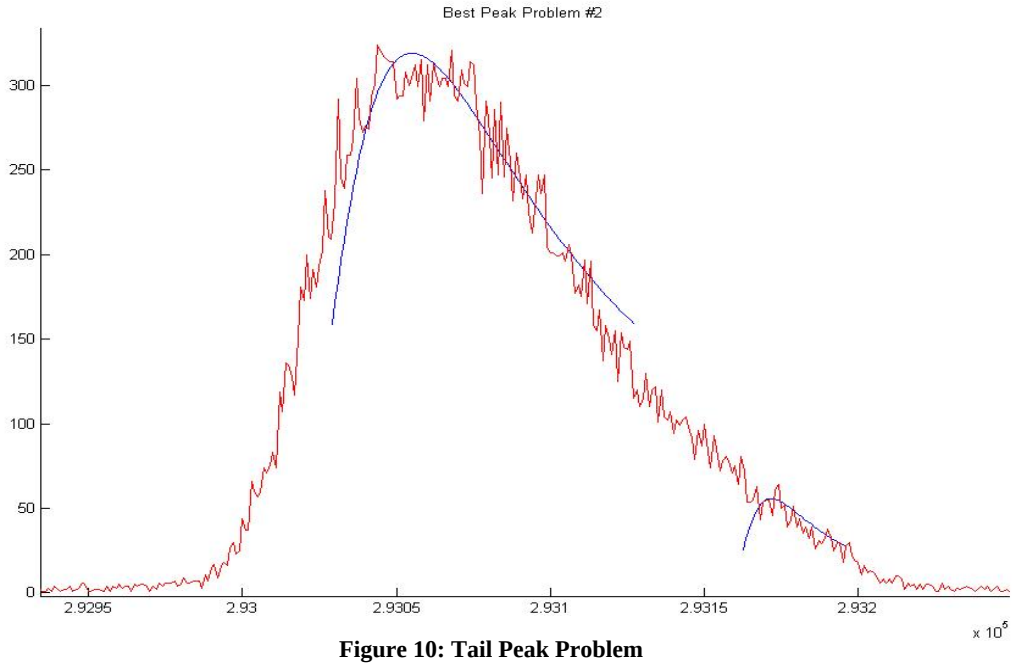

Figure 10: Tail Peak Problem

window around the actual peak been larger, the tail peak would have been considered the same peak with an obviously worse error and thus be eliminated. However, since the window is too small, this peak escapes deletion. The first obvious solution is to increase the window size. Unfortunately this simple solution can have undesirable consequences as seen in figure 11. If we increase the window around the left peak too much to include the right peak, then peak picker will eliminate one of two very real peaks, an obvious problem.

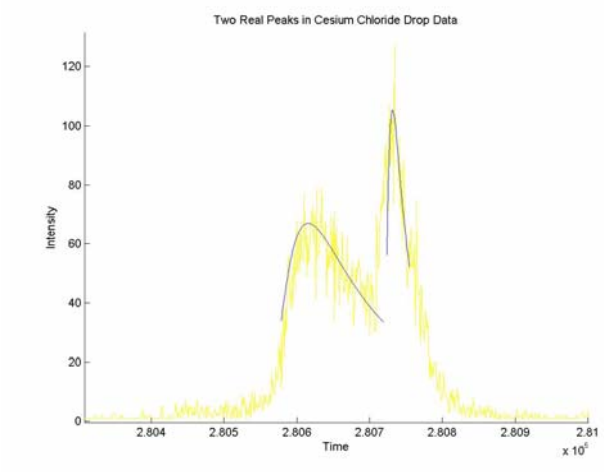

Figure 11: Two Real Peaks which would be affected by first solution

Instead, to fix the problem I wrote a new program called Sremovebad.m (Appendix A.4). This problem runs through a peak list (like the one produced by the bestpeak program) and searches for peaks which are close together. It does

this by extending the window around the peak to being five times their width, an increase which would include all tail peaks and peaks like the right peak in figure 11. Then, if the amplitude of the second peak is less than 1/5 of the amplitude of the first peak, the second peak is removed (the 1/5 factor was chosen after examining the amplitude of the typical tails). This effectively removes tail peaks, without removing peaks like figure 11. This process sometimes requires multiple runs until all tail peaks are removed, thus Sremovebad2.m (Appendix A.5) was written which continually runs Sremovebad.m until no more peaks are being eliminated. Figure 12 shows before and after pictures of the process being run on the CsCl droplet data. With this final step complete, our efficient and effective process of identifying a large number of different peaks (or different shapes and sizes) was complete.

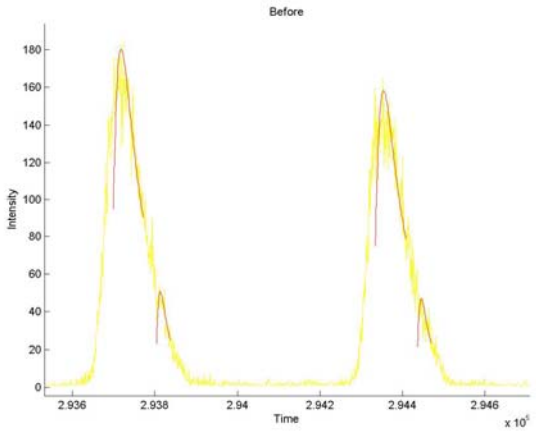

Figure 12a: Before Sremovebad2

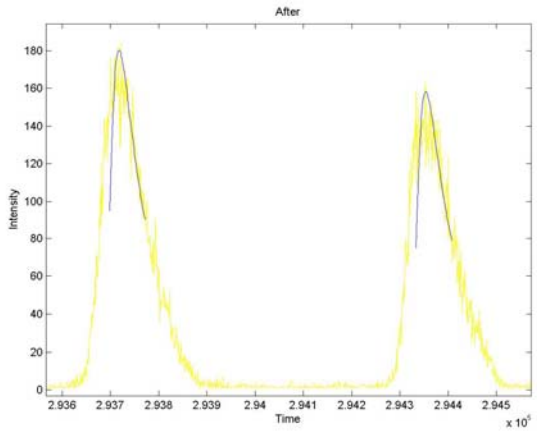

Figure 12b: After Sremovebad2

2. Detection and Analysis of Global Position Dependent Peak Shifts

a. Evaluation of a Single Mass Peak

Now that we could successfully reduce our data, the next step was to begin searching for time shifts between the same peaks in different regions of an image. At this point I began shifting my attention away from the CsCl droplet data, and began examining an image of a biomotor etched in a silicon substrate (approximately 200 microns across). The full image being

considered can be seen in figure 13a. The first step in detecting any shifts was to choose the two regions which would be considered (figure 13b) and obtaining the spectrum and peaks lists from

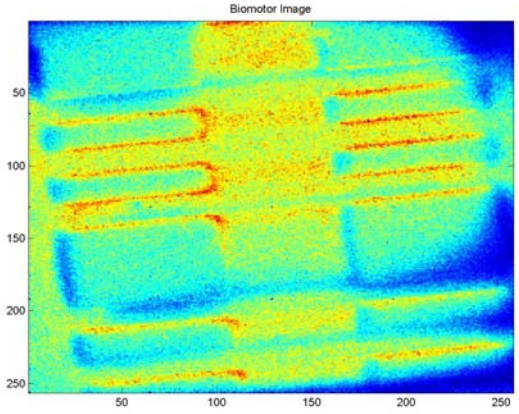

Figure 13a: Biomotor Image

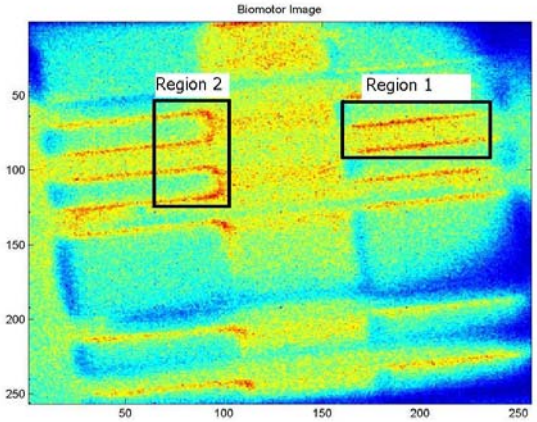

Figure 13b: Biomotor Image Regions

these regions using the best peak program. After investigating the benefits of running the programs with a variety of different parameters, it was found that using resolutions of 4000 and 7000 provided the best results (i.e. fits with low errors for the most peaks). I was now ready to search for any shifts between the two regions.

Initially my goal was simply to choose one peak and manually detect the shift between that peak in both regions. Once I knew the shift, I could then adjust the data from the second region so that the peaks from both regions aligned and recombined the data with an improved resolution for that peak. To do this I randomly chose a peak (mass = 126 Daltons), and found the difference between the peak position from region 1 to region 2 as found by the best peak program. I found this difference to be 23 clock ticks (where one clock tick is 0.138 ns). I then aligned region 2 to region 1 by simply adding this time difference to the time of each detection from region 2. Finally, I calculated the spectrum from the combined original detections, and the spectrum from the combined aligned detections to see the improvement.

Figure 14a demonstrates the alignment of the individual region peaks. In the figure, the black peak represents the unchanging region 1 peak, the green line represents the peak from

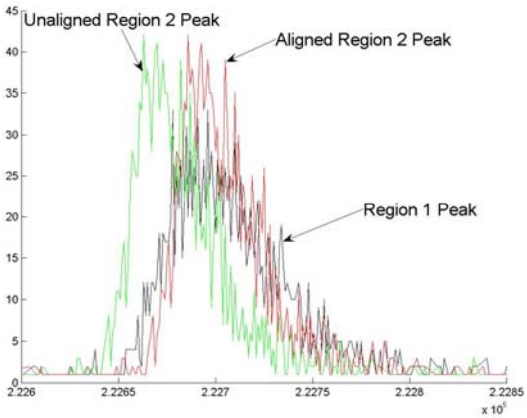

Figure 14a: Realignment of Individual Region Spectra

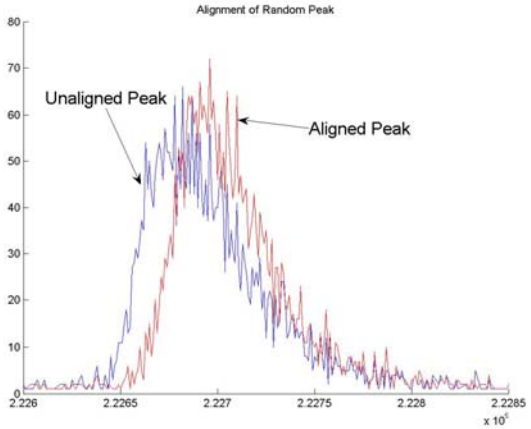

Figure 14b: Realignment of Combined Spectra

region 2 as it originally appears in the data, while the red line represents the peak from region 2 after it has been aligned. Noticed how the red peak appears in the same time position as the peak from region 1. Figure 14b demonstrates the change from the combined spectrum from regions 1 and 2 before alignment (blue) and after alignment (red). Note how the aligned peak is slightly narrower and has a higher amplitude, indicating a slight improvement of resolution (a more detailed discussion of what classifies as improvement will follow).

b. Characterization and Removal of a Global Shift

The next step in detecting the shifts between two regions was to examine how the shift varied with time. For example, if we found that the shift was constant with time we could simply shift all of the detections in one region by a constant amount (as we did previously) and obtain our new aligned spectra. To investigate this change with time I wrote a program (Sshfit.m, Appendix A.6) which could examine the peak lists from two regions, and using a process similar to the bestpeaks programs, identify common peaks and calculate the shift that exists between

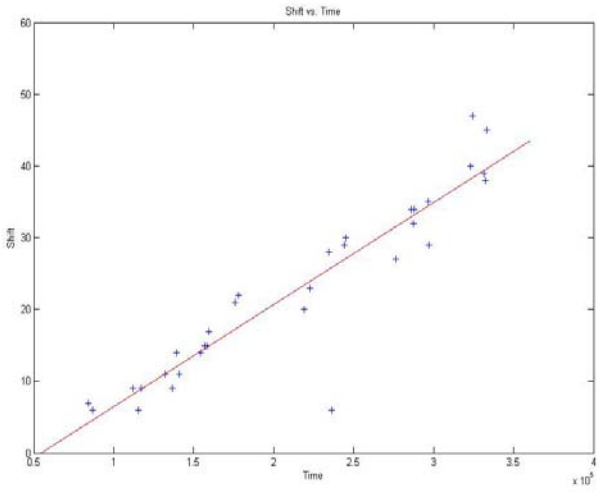

Figure 15: Change in Peak Shift with Time

their positions. I then ran this program on the entire spectrum for the biomotor data and obtained the results seen in figure 15. With the exception of one odd peak we can see that the shift between two common peaks depends linearly on time. Therefore, to determine how to shift the detections I simply used MatLab to find the best fit straight line to the shift results, and adjusted the detections according to this dependence.

To this effect, I wrote another program (Salign.m, Appendix A.7) which takes the shift data calculated by the Sshift program, along with the detection data from the two regions being considered, and produces three new spectra: the new spectrum of the shifted region, the

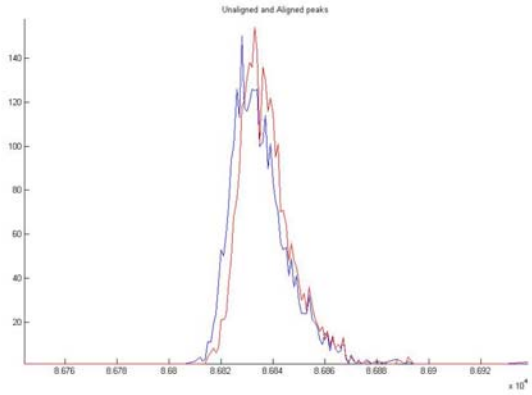

Figure 16a: Aligned Peak Earlier in Time

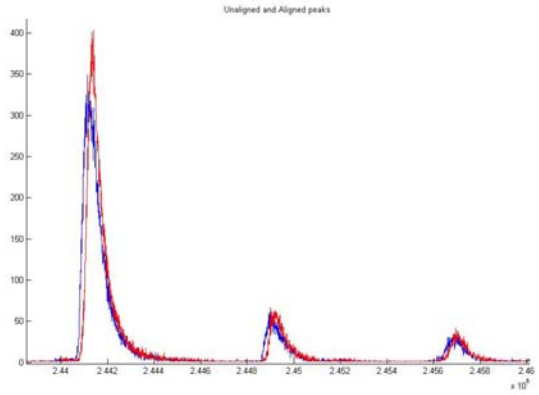

Figure 16b: Aligned Peak Later in Time

combined spectrum of the original regions, and the combined spectrum of the new aligned regions. Figures 16a and 16b show how the combined spectra (blue) from region 1 and region 2 are aligned (red) using these new programs to produce a new aligned peak at two different times

in the spectrum. One can clearly see a more drastic change in 16b, which is to be expected because this peak exists further along the time axis, where the shift between the two regions is greater and thus the effects of alignment is increased. At this point, a more detailed discussion of what is meant by improvement is required to determine whether any true effect is being seen.

When the resolution of a particular peak is increased, there are several quantitative ways we can expect to see the improvement. The first, and most obvious, is that we expect to see the width of an aligned peak to be narrower then the unaligned peak. As a result, since the number of detections does not change, we expect to see the amplitude of the aligned peak to be greater than the unaligned peak. Therefore, examining the change in full width half maximums between our aligned and unaligned peaks would reveal whether resolution was being enhanced.

Furthermore, another test as to the effectiveness of our alignment is to see whether our original predicted line shape describes our new aligned peak better than the unaligned data peaks. This occurs because our original line shape does not take into account distortions caused by sample variations. Therefore, these variations tend to reduce the accuracy of our predicted line shape. If we are correctly accounting for these variations, then we would expect the success of our predicted model to increase. This can quantitatively be examined by evaluating how the error produced by the peak picker program changes between the aligned and unaligned peaks. If the error is less for the aligned peaks, then our aligned data is better described by our predicted model, indicating improvement in resolution.

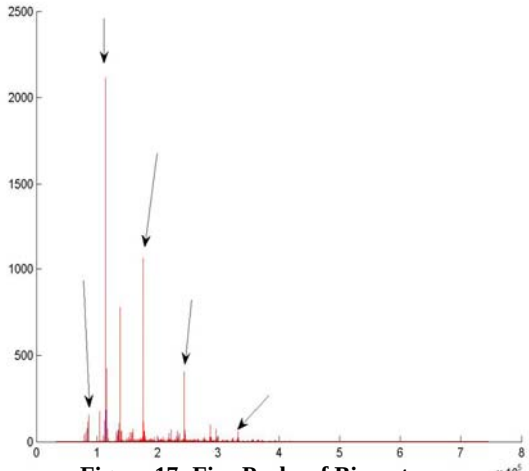

Figure 17: Five Peaks of Biomotor Spectrum to be analyzed

To test the actual effectiveness of the alignment being conducted, I chose five different peaks from the biomotor spectrum, and determined how our two indicators (full width half maximum and peak picker error) changed. Figure 17 shows which five peaks I chose from the biomotor spectrum while figures 18a and 18b show how the two indicators change between the aligned and unaligned peaks. For all five peaks a decrease in full width half maximum was seen and figure18a shows the percentage decrease of the original full width half maximum. As

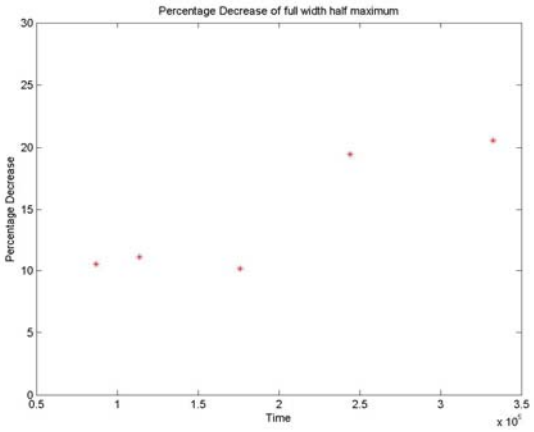

Figure 18a: Percentage Decrease in Full Width Half Maximums

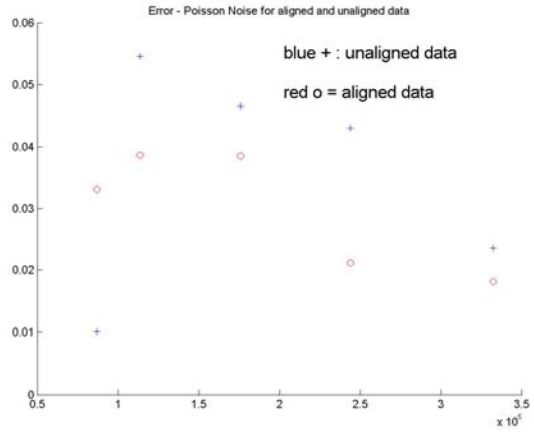

Figure 18b: Change is Error of Biomotor Spectrum Peaks

expected we see the effect is greater for the peaks which exist where the shift is greater (later in time). Figure 18b shows the change between the aligned peak's error (blue '+') and the unaligned peak's error (red 'o'). With the exception of the first peak, we do see improvement, however, it is important to note that the errors for these peaks are already quite low for the unaligned peaks, therefore, this may not be the best indicator for this particular data set. Ideally, to really test the effectiveness of these alignment procedures, a data set with clear shifts between different regions of an image would be analyzed. Luckily, such a data set was available for me to investigate.

c. Initial Investigation of Rtil Drop Data

After using the biomotor data to create the shift detection and elimination programs, I then moved on to a data set with very obvious shifts which I could attempt to remove. The new data was an image of two RTIL droplets (a conductive solution) on a silver coated silicon substrate. An image of the new data set can be seen in Figure 19a. This initial image shows all ion detections, while figure 19b show only detections around our known PF_6 peak (a known ion

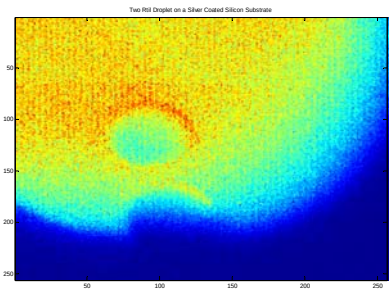

Figure 19a: Rtil Droplet Total Ion Image

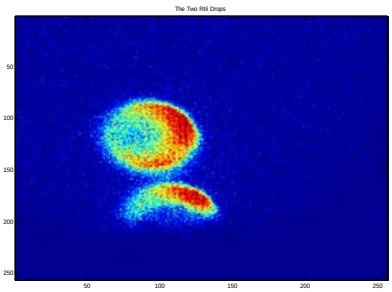

Figure 19b: Rtil Droplet PF_6 Ion Image

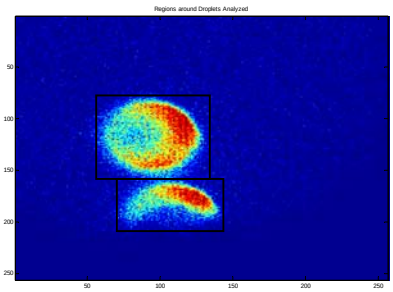

Figure 19c: Rtil Droplet Analysis Zones

in the Rtil solution) so the two drops can clearly be seen (both drops are about 50 microns in diameter). For my purposes, my attention was focused on the two drops so I created two regions around these drops as seen in the figure 19c. By examining the spectrum created by the combined detections from the regions (figure 20a) an obvious problem can be seen. The shift

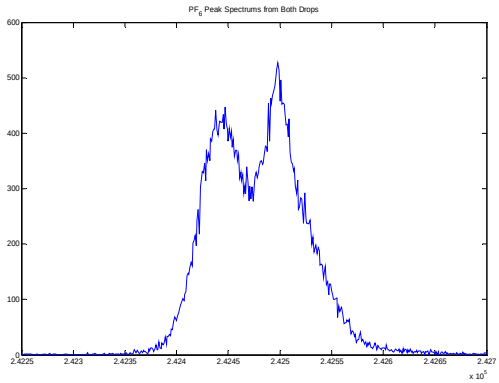

Figure 20a: Combined PF_6 Peak from Both Drops

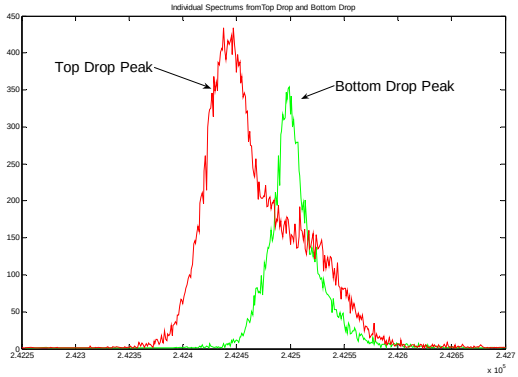

Figure 20b: Individual PF_6 Peaks from Both Drops

between the two drops is so great, that the single PF_6 peak appears as two peaks in the combined spectrum for the droplet detections. However, when examining the two regions separately (figure 20b) we can see that one peak does in fact exist in each region but they are shifted considerably from one another.

The alignment process of this peak was fairly straightforward. I first reduced the spectrum so that I was only looking at the PF_6 peak, then I ran the alignment programs I had developed on the biomotor data. Figure 21a shows the alignment of the drop 2 peak (green = unaligned, red = aligned) with the drop 1 peak (blue), and figure 21b shows the combined

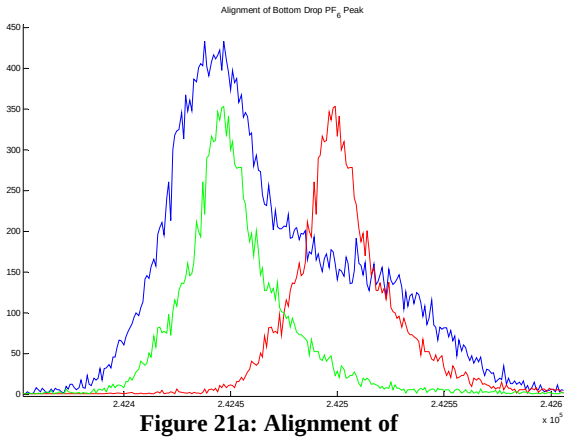

Figure 21a: Alignment of PF_6 Peaks between Drops

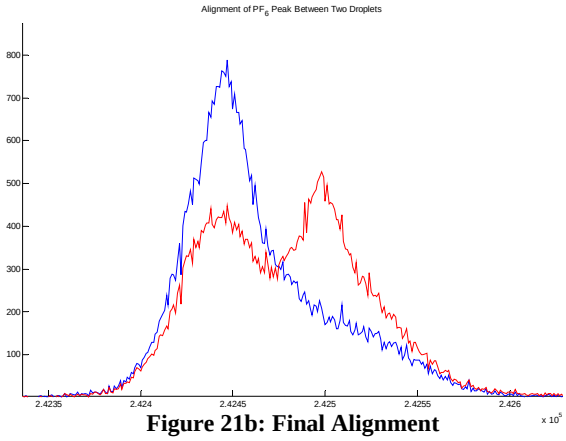

Figure 21b: Final Alignment of PF_6 Peak

spectrum of the unaligned regions (red) and the spectrum of the aligned regions (blue). After running the alignment procedures we can now see that our previously unresolved peak has become resolved as a single peak, indicating a significant improvement of resolution, and the success of my alignment procedures. Nevertheless, a tail still exists within the blue aligned peak, hinting that better alignment is possible.

3. Alignment through Modeling of Complex Surface Structure

a. Feature Specific Investigation of PF_6 Peak

At this point in my research the methodology I was using to aligned the data changed significantly. Rather than attempting to detect the shifts of all the peaks in a data set (as I did for the biomotor data), I began to characterize the time shift of a sample using just a single ion. Then, by scaling the shifts accordingly, I hoped to align the other peaks.

In an attempt to apply this new technique, I first set out to understand the shifts which exist on the PF_6 peak within the top droplet region. To do this, I reduced the total number of detections within the data set to just those ions which fell both within a small window around the PF_6 peak and within the region of the top drop. The combined spectra from all of these detections can be seen in figure 22. I then divided the region into equally sized subregions and found the individual spectra for each of these regions. I performed this process using a program I had written (SspecChange.m, Appendix A.8). For example, if I divide the region into 16 subregions, figures 23a and 23b show four of the individual spectra from these

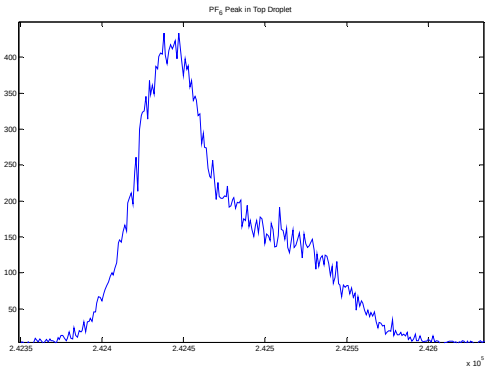

Figure 22: PF_6 Peak From Top Drop

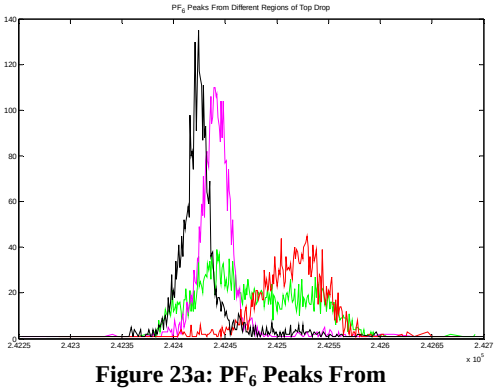

Figure 23a: PF_6 Peaks From Different Regions of Top Drop

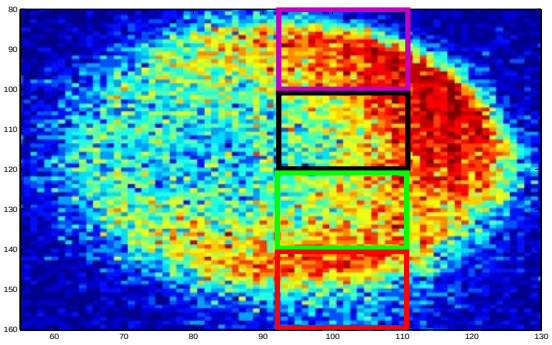

Figure 23b: Corresponding Regions of Figure 30a

different regions of the droplet data. The specific color of a peak in figure 23a corresponds to the same color's region in figure 23b. We can see that the size, shape, and position of the peaks can change quite drastically between regions. In particular, we can see that peaks coming from the bottom of the droplet, which are wider and have lower amplitudes, are combining to create the unusual shape we see in the decline of the peak in figure 22. More importantly, however, these spectra show that the distortion in the total spectrum is the result of combining shifted spectra from different regions (i.e. inhomogeneous broadening), and not the result of some fundamental broadening from the apparatus. Therefore, we can improve the resolution of the drop by removing this spatial dependent shift

The original hope was that I could apply the same alignment techniques as before on a smaller scale to align each of the individual regions' spectra. Unfortunately, the number of ion counts which exist within each of these subregion is considerably lower then any ion count which had previously been investigated, especially around the outside of the droplet. This caused significant problems with the peak picker, which ultimately forced us to abandon it and attempt to quantify the spectrum peaks' positions in new ways. Consequently, I began examining the centroids and medians of the different peaks.

The centroid of a peak is essentially equivalent to its center of mass, but instead of using mass we use ion count. To calculate the centroid of all the peaks in a spectrum, I wrote a program (Scentroid.m) which utilizes equation 8 to find the centroid of each peak. In this equation, x_i refers to the position of a point, $A(x_i)$ refers to the ion count at that point, and the summation is performed over an entire peak.

$$Centroid = \frac{\sum x_i A(x_i)}{\sum A(x_i)} \tag{8}$$

The other feature, the median of a peak, is defined as the position in which half of the total area under the curve (in this case, half the number of ion counts) is to the left, and the other half is to the right. To investigate this feature I wrote a program (Smedian.m, Appendix A.9), which would calculate the medians of all the peaks in a spectrum which is inputted.

I then wrote a program (Scent.m), which would calculate the shift between the centroids of each region and the region with the centroid position which was furthest in time. I also wrote a similar program (Smed.m, Appendix A.10), which would do the same thing for the medians. Figures 24a and 24b shows how the shifts of the centroids and medians change as a function of position when the data is dividing into 225 regions (note in each image one region has a shift of

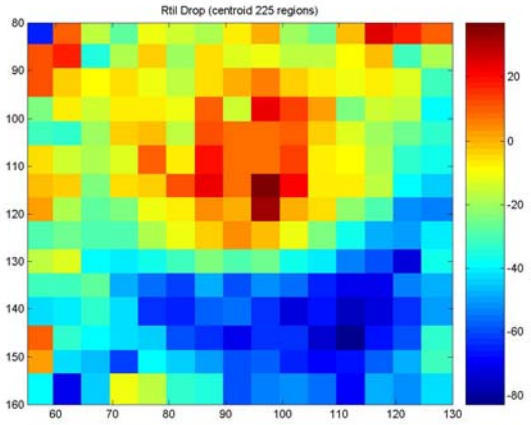

Figure 24a: Centroid Shifts Between PF_6 Peaks From 225 Different Regions of Top Drop

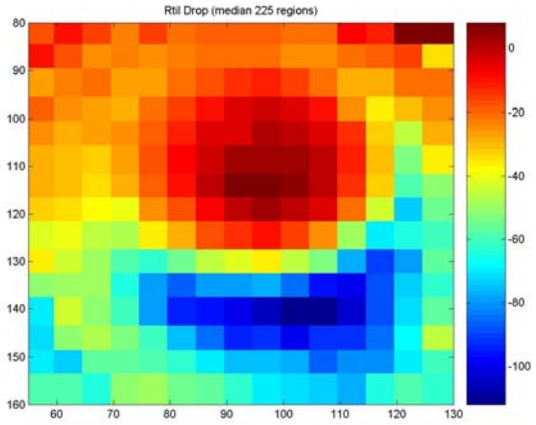

Figure 24b: Median Shifts Between PF_6 Peaks From 225 Different Regions of Top Drop

zero, this is the region from which every other shift is calculated). When examining these images we can clearly see that the centroid shift image is significantly noisier than the median shift image. This is a reasonable effect when we consider that the centroid of a peak is going to place more importance on the tail features of our peaks (i.e. when x_i is large) than the median. Since these tails features will be most affected by the noise in the data, the noise will be most significant on the centroid. Therefore, from this point on, I decided to focus purely on the median shifts of the peaks, and ignore the centroid shifts.

After deciding on the median as our choice for peak position determination, I had to determine how small I could allow the region to be and while still obtaining usefully information. Ideally we would be able to divide the region into as many pixels as it contains (6156), but realistically, at such small ion counts, we could no longer trust the median shift information coming from the programs I had written.

After examining the median shifts for a variety of different region sizes, I determined that dividing into 324 regions provided the most useful information, without allowing the corner shifts to be too extreme and negatively affect the outcome of the alignment process. Figure 25 shows the median shifts between 324 regions of the top drop data. While large shifts can still be seen in the far corners, these shifts are comparable in size to the largest shifts seen in the larger ion count regions and should not influence the alignment to a large degree.

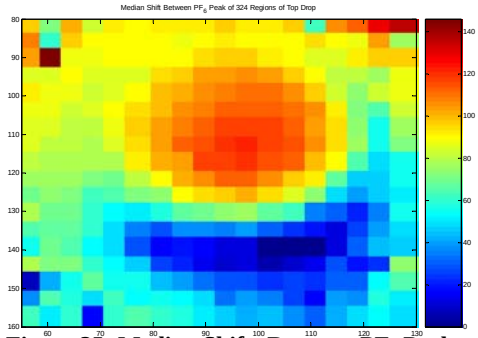

Figure 25: Median Shifts Between PF_6 Peaks From 324 Different Regions of Top Drop

The next step towards aligning the PF_6 peak with the top drop region was to actually use the median shift information I had obtained to shift the different spectra appropriately and recombined them. I wrote a program (Smedalign.m, Appendix A.11) which would perform this task. At this point our focus has shifted from aligning all of the peaks, to simply aligning one peak. Therefore, the Smedalign.m program requires the input of a window around the peak to be aligned. The program then collects all the relevant detection from within each subregion, and shifts them according to the median shift calculated for that subregion. Figure 26a shows both the new aligned spectrum (red) and the old unaligned spectrum (blue) of the PF_6 peak in the top

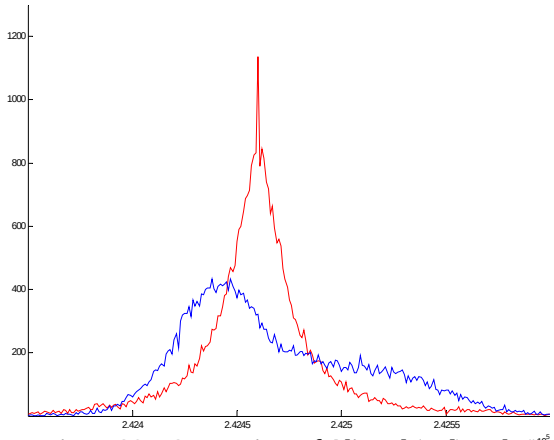

Figure 26a: Comparison of Aligned (red) and Unaligned (blue) PF_6 peaks from Top Drop

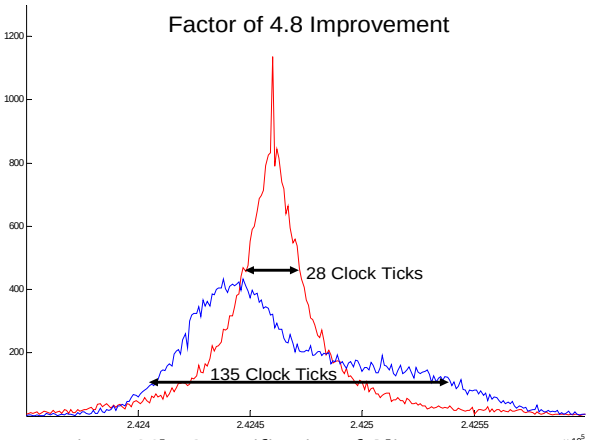

Figure 26b: Quantification of Alignment Success of PF_6 peaks in Top drop

drop. Clearly a significant decrease in the width of the peak can be seen. Additionally, the unusual gradual decline of the peak has been replaced with a faster decline which we would expect.

In order to actually characterize the improvement we see between the aligned and unaligned spectra, in figure 26b I evaluated the width of both peaks. In these plots, a clock tick is 0.138 ns. Through accounting for the shifts in the top drop region, we see an improvement by a factor of 4.8 in the resolution of the PF_6 peak.

b. Overall Resolution Improvement of PF_6 Peak in Rtil Drop Data

We have already seen that through comparing the shift between the top and bottom Rtil drops, we can align the PF_6 peak to remove the presence of two false peaks, and replace it with the true peak. Additionally, we can improve the resolution of the top drop peak by a factor of 4.8 by accounting for the shifts which exists only within the top drop region. The final step in improving the resolution of the PF_6 peak is to combine these two steps. Since we had already performed the alignment of the top drop based on its regional shifts, I next had to align the bottom drop based on its region specific shifts using the same steps we did for the top drop.

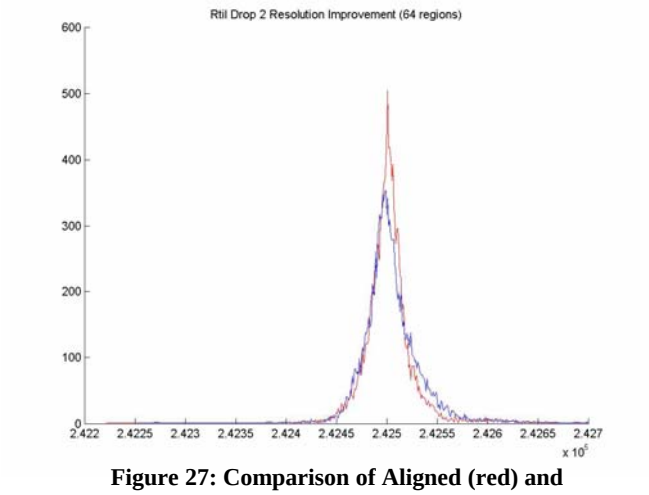

Figure 27: Comparison of Aligned (red) and Unaligned (blue) PF_6 peaks from Bottom Drop

Figure 27 shows the improvement of the bottom drop's PF_6 peak which results from removal of the region specific shifts in the data. It is quite obvious that the improvement in the bottom drop's peak is much less than that of the top drop. This is true for several reasons. First, only the top half of the bottom drop was analyzed in the data. This means there is a significantly lower ion count which reduces the number of regions which can be used in the detection of the median shifts. With fewer regions, our improvement capabilities are lower. Furthermore, since the bottom drop has a higher initial resolution, improving the resolution further is more difficult. Nevertheless, some improvement is seen, and we are now ready to align the two regionally aligned peaks.

The final alignment of the two peaks was performed using the same peak picker process as before within a small window around the PF_6 peak. Figure 28a shows the resolution

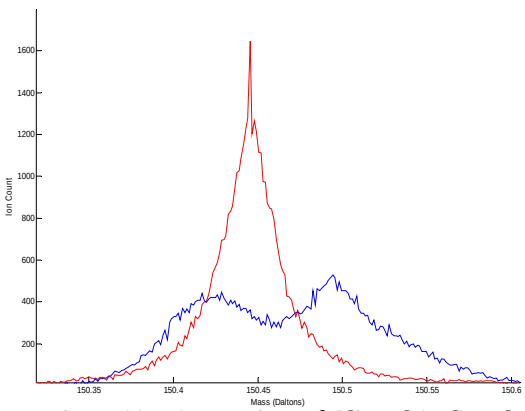

Figure 28a: Comparison of Aligned (red) and Unaligned (blue) PF_6 peaks (All Detections)

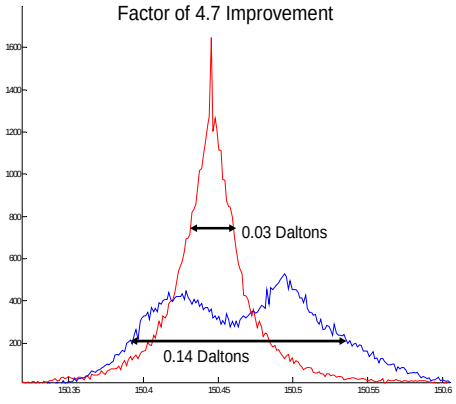

Figure 28b: Quantification of Resolution Improvement between aligned (red) and unaligned (blue) PF_6 peaks (All Detections)

improvement of the PF_6 peak and figure 28b quantifies this improvement by the same process used earlier. Through utilizing these two techniques we see the PF_6 peak from the original data, which consisted of a wide mass range and the appearance of two separate peaks, reduce to a mass range almost five times narrower with only one well defined peak. This represents the current ultimate resolution improvement possible for an individual peak utilizing my alignment techniques.

c. Alignment of F Peak Using PF_6 Shift Data

If the ultimate goal was to improve the resolution of a single peak, then using the techniques I have outlined would represent the end point in resolution improvement. Ideally, however, we can use the shifts from a known peak (such as our PF_6 peak) to align other peaks (such as the Fluorine (F) peak in the Rtil droplets). Therefore, I continued my analysis of the top drop in an attempt to determine how to align ions of different masses with the known shifts of one ion.

The next step was to find a new ion which could also be analyzed within the top drop. Luckily, the Rtil droplets also contain a large amount of the negative F ion. Therefore, I collected the detections which fell within the top region and within a window around the F peak.

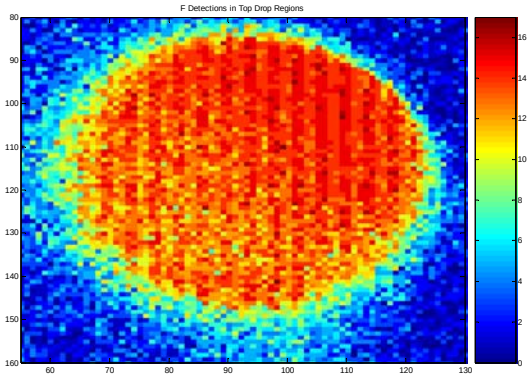

Figure 29a: Total Ion Image of F Detections in Top Drop

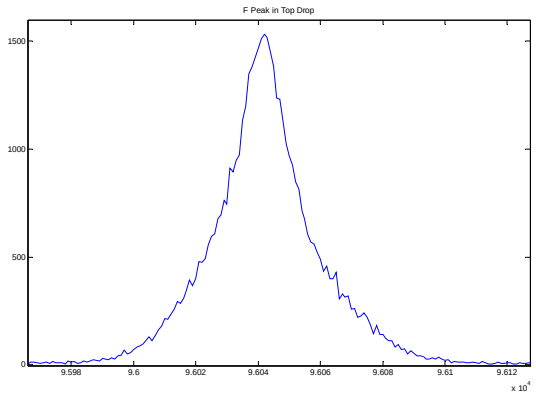

Figure 29b: Spectrum of F Detections in Top Drop

Figure 29a shows the total ion image of all these detections while figure 29b gives the spectrum of the detections. I then followed the same techniques I used to align the PF_6 peak for the F peak using 121 regions. Figure 30a shows the median shifts found using the F peaks and figure 30b shows the resolution improvement found using this shift data. Note that I changed the region from which all other median peaks were compared. For clarity purposes median shifts are now detected from the median peak earliest in time, and not latest.

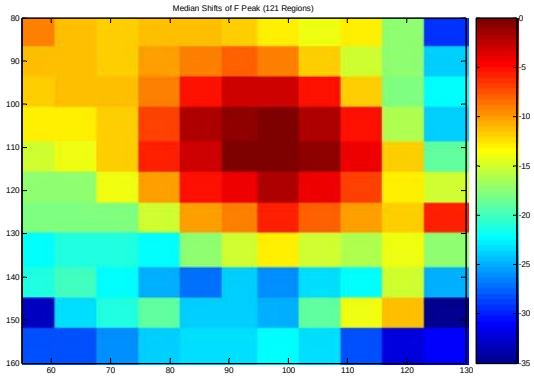

Figure 30a: Median Shifts of F Peak

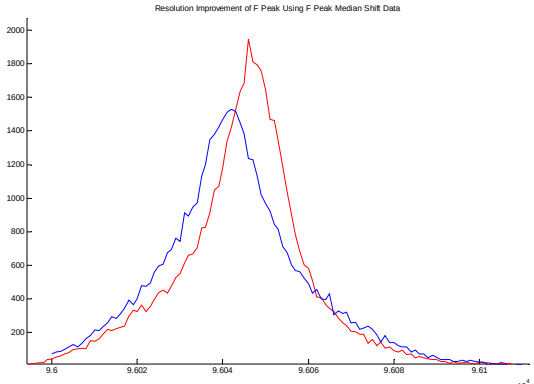

Figure 30b: Comparison of aligned (red) and unaligned (blue) F peaks

We can see that the improvement of the F peak is not as drastic as it was for the PF_6 peak. However, we must remember that the initial resolution of the F peak was superior, reducing how effective alignment can be.

Now that we have seen how effective aligning the individual regions of the top peak using the F median shift data can be in improving resolution of the F peak, we must also see how effective the alignment process would be using the PF_6 median shift data. To do this, we must first remember that the shifts on the F ions are not as significant as they are on the PF_6 ions, therefore we must scale the PF_6 shifts accordingly. Our previous experience showed us there was a linear dependence on time for shift on an ion. Since the time of flight of an ion is linearly dependent on its velocity, which is proportional to the square root of its mass ($mv^2/2$), our initial guess was that our scale factor would be equal to the square root of the ratio of the ion masses

(19 daltons for F, 145 for PF_6, ratio ≈ 3). I then adjusted the Smedalign.m program to allow for the input of this scale factor. Figure 31 shows the alignment results using both the F median shifts (red) and PF_6 median shifts (black) (the unaligned peak is blue). Clearly the PF_6 data does almost exactly as well as the F data in aligning the F peak. In fact, the PF_6 data does a better job of pulling in the tail of the F peak.

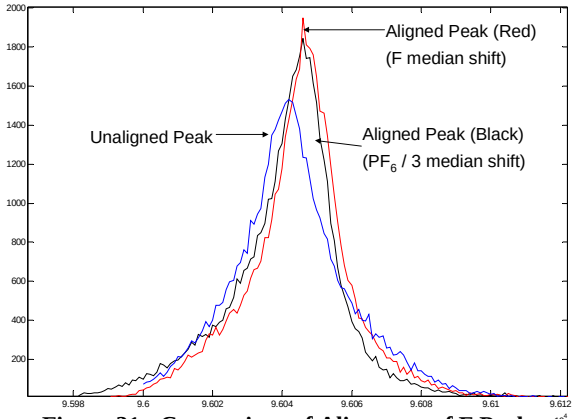

Figure 31: Comparison of Alignment of F Peak in Top Drop Using PF_6 Median Shift (black) and F Median Shift (red) [unaligned in blue]

By successfully showing that the PF_6 data can be used to align the F peak I have shown that we can use the median shifts of the PF_6 peak to align other peaks within the same region. Ideally, I would have like to examine another unknown ion within the Rtil drop to see if alignment could also be improved on it. Unfortunately, no such ion existed in the data. Instead, I focused my attention on attempting to do the opposite of my previous example. Rather than aligning the F peak with the PF_6 median shift data, I attempted to align the PF_6 peak using the F median shift data. Figure 32 shows the alignment attempts on the PF_6 peak using the PF_6 median shift data (black) and the F median shift data (red) (the unaligned peak is in blue). Quite clearly we see that the F data does not do as good of a job as the PF_6 data.

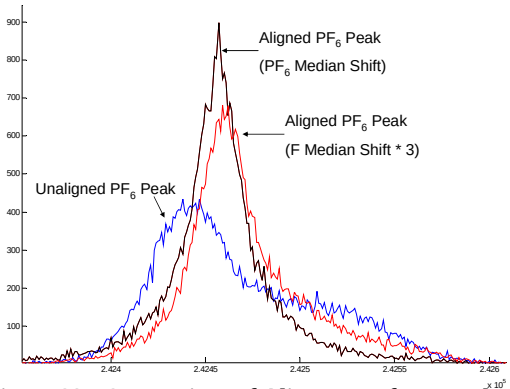

Figure 32: Comparison of Alignment of PF_6 Peak in Top Drop Using PF_6 Median Shift (black) and F Median Shift (red) [unaligned in blue]

This failure presents a problem because, ideally, either ion's data could be

used to align the other ions in the region. Understanding why this difference exists was crucial to applying the overall alignment technique in the future. To discover the cause I had to examine the shifts in the region more closely and attempt to discover what was causing them.

d. Investigation into the Cause of Median Shifts

The most obvious possible cause for the median shifts experienced in the top drop region is the droplet itself. If we assume the droplet to be a sphere with a diameter of 50μm, then we know the difference in sample height between an ion on top of the droplet and an ion outside of the droplet is 25μm. This difference represents an extra distance the non-droplet ions must traverse before reaching the detector, which may account for the shift. Within the apparatus, we assume all ions gain or lose only one electron and therefore obtain a charge of ±1. After traveling through a 3 kV potential, all ions obtain an energy of 4.8×10^{-16} Joules ($E = qV$). Using the PF_6 ions which have a mass of 145 Daltons (2.42×10^{-25} kg), this translates to a final velocity of 6.30×10^4 m/s ($E = mv^2/2$). From this point on the ion is not accelerated and all ions travel the same distance in the same time. However, while reaching the final velocity, some ions have traveled farther. The average velocity of the ion over its acceleration time is half the final velocity or 3.15×10^4 m/s. This means that since the non-droplet ion must travel 25μm further while accelerated, it experiences a time shift of 0.79ns. Since each clock tick in the TOF-SIMS apparatus is equal to 0.138 ns, this converts to a time shift of 5.75 clock ticks

Figure 33a shows the PF6 ion image of the top drop while figure 33b shows the median shift calculated using the PF_6 peak when divided into 225 regions (note that these figures are on the same spatial scale). Even if we take the smallest shift between the region on top of the drop

and the region outside the drop we obtain a shift on the order of 20 clock ticks (in the same direction). Clearly the droplet's surface height changes can not be the sole cause of the shift.

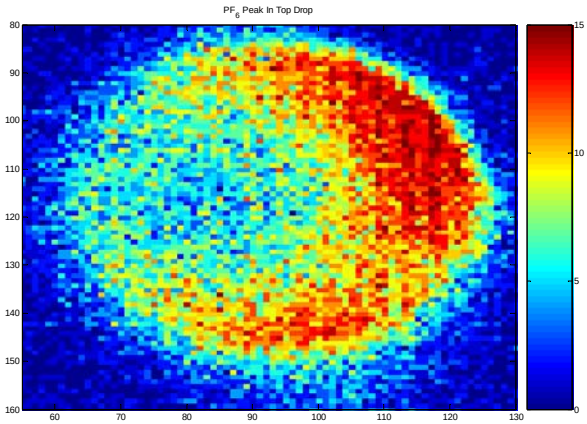

Figure 33a: PF_6 Ion Image of Top Drop Region

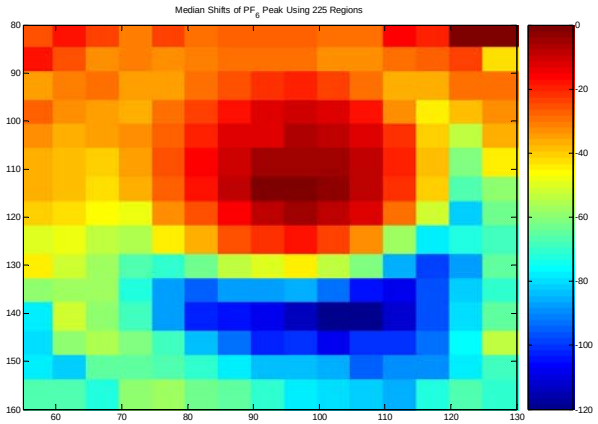

Figure 33b: Median Shifts of PF_6 Peak in Top Drop Using 225 Regions

There was one major flaw in our calculation of the time shift experienced by an ion near the droplet. We assume that immediately after an ion is ejected from the droplet it enters the electric field of the TOF-SIMS apparatus and is accelerated towards the detector. This assumption does not hold if the Rtil droplet is shielding the ions from the electric field of the apparatus. This is not unreasonable since Rtil is essentially composed of mobile cations and anions which will distribute themselves within the droplet to prevent the apparatus's electric field from penetrating the droplet.

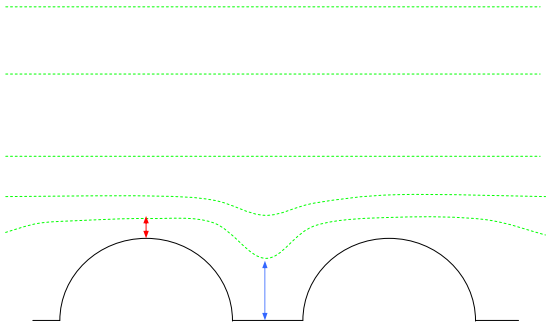

Figure 34: Visualization of Shielding Effect of Rtil Droplets

Figure 34 represents this effect. The black curves represent our droplets which we assume to be equipotential because of the mobility of the ions within the Rtil. The green dashed lines represent equipotential lines. As the straight lines begin to approach the Rtil droplets they begin to deform to match the

curves equipotential of the curved surfaces. The end result, however, is that the distance from the surface to the first equipotential line is significantly less on top of the drop. This means the electric field is much stronger. Between the droplets, the shielding effect of the droplets is reducing the electric field. The end result is that it takes ions from this region longer to reach their maximum velocity, and this produces the time shift we see in those detections.

Using this explanation we can understand why the time shift may be greater than that expected from just the droplet geometry. If the explanation about the electric field shielding of the Rtil droplets it correct, then the bottom drop would in fact be shielding the top drop's bottom half. With this increased shielding from the bottom drop, we would expect to see the increased time shift of the bottom half of the top drop. To investigate this effect more closely, I turned to the PF_6 detections around both drops.

Figure 35a shows the PF_6 ion image of a large region placed around both droplets while

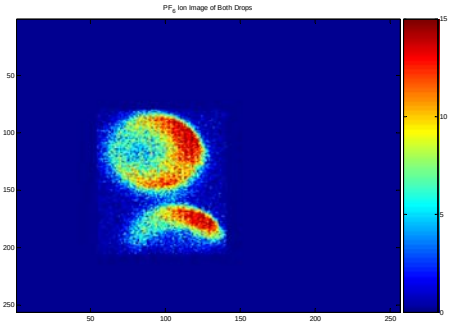

Figure 35a: PF_6 Ion Image of Both Drops

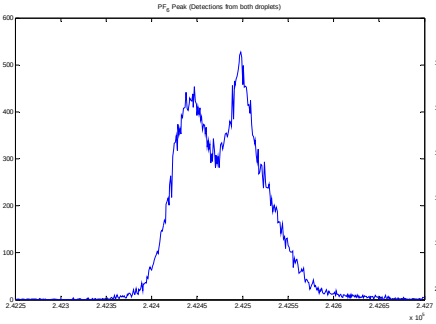

Figure 35b: PF_6 Peak Combined From Both Drops

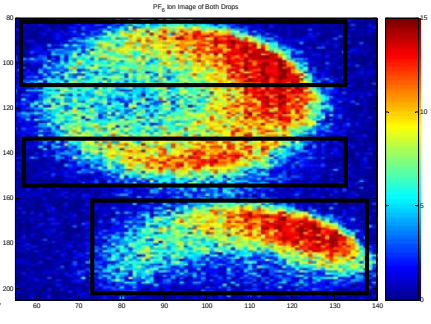

Figure 35c: New Region Specific Detections

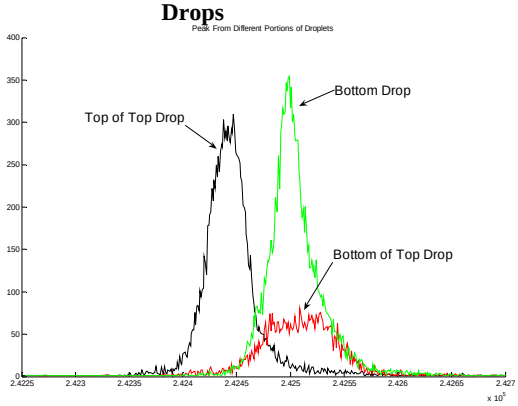

Figure 36: PF6 Peaks From 3 Regions of Figure 45c

figure 35b shows the spectrum of the relevant detections within this region. On the other hand, figure 35c shows the PF_6 ion image divided into three new regions; the top of the top drop, the bottom of the top drop, the bottom drop. If our

assumption is true, and the two drops are shielding the zone between them from the electric potential, then we would expect the peak from the bottom of the top drop to align more with the bottom drop peak then the peak from the top of the top drop. This is exactly the effect we see in figure 36 which shows the spectrum of the regions from figure 35c.

In light of this alignment, it seems likely that the reason the peak from just the bottom drop (figure 43 red) is not initially shaped like the peak from the top drop (figure 43 blue) is that it does not have a bottom region which is further from the shielded zone. Without these less shifted detections to pull the peak to the left the peak becomes more resolved (in the shifted position).

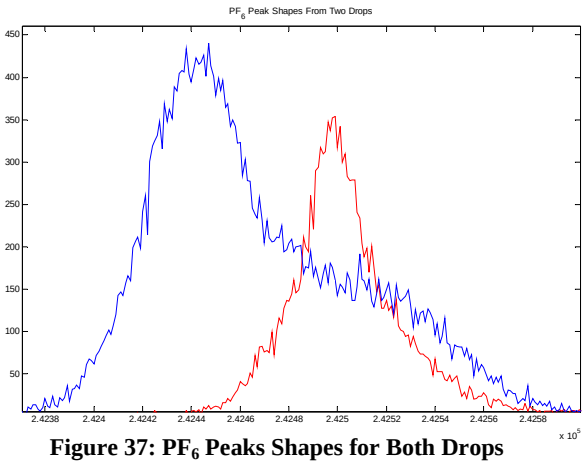

Figure 37: PF_6 Peaks Shapes for Both Drops (Top Drop = blue, Bottom Drop = red)

These images provide strong evidence in support of the assertion that Rtil electric field shielding is the main cause of our time shifts. Unfortunately, they do not solve our original problem as to why the PF_6 median shift data was better at aligning peaks than the F median shift data. To answer this question, we must investigate the effects this shielding has on the F peaks.

I performed the same techniques on the F peaks and obtained similar images. Figure 4a shows the F Ion image of the detections from both droplets, figure 44b shows the combined F peak from the droplets, figure 44c shows the new regions which will be examined, and figure 44d shows the F peaks from the different regions. Once again we see a slight appearance of a second peak in the combined F peak. Furthermore, again we can see that the bottom portion of the top drop aligns with the bottom drop, rather then the top portion of the top drop. As

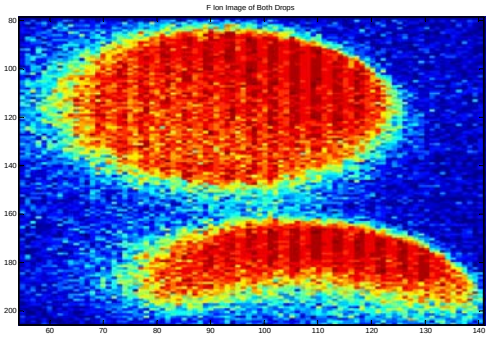

Figure 38a: F Ion Image From Both Drops

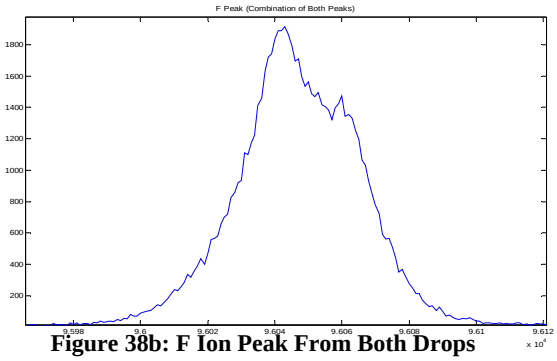

Figure 38b: F Ion Peak From Both Drops

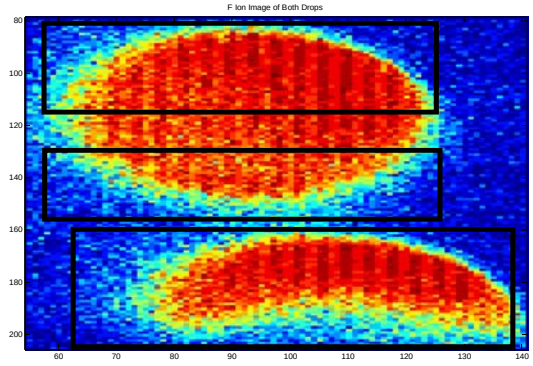

Figure 38c: F Ion Image Regions

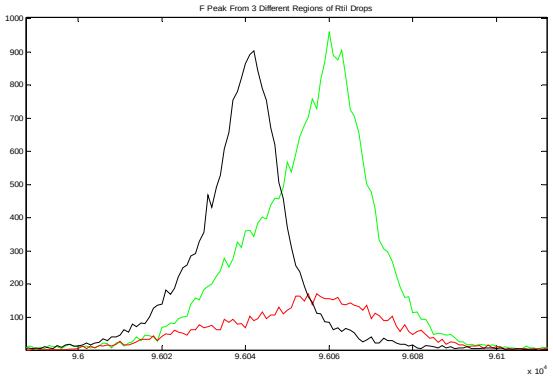

Figure 38d: F Peak From F Ion Image Regions

expected, the shielding effect also appears in the F shifts, but to the lesser degree due to the F ion's lower mass. However, we must examine one more feature to see why the PF_6 median shift data provides better alignment then the F median shift data.

It is obvious that there are detections from both the PF_6 and F peaks which get shifted significantly by this shielding effect, but what is not immediately obvious is what percentage of the total detections for a given peak within the top drop region are really being shifted significantly. If we look closer, we will see that a much larger percentage of the PF_6 ions are being shifted significantly than the F ions. Figure 45a shows the F peaks for two regions; one including the entire top drop (blue) and one including the bottom portion seen previously (red). Figure 45b shows the PF_6 peaks for the top drop region (blue) and a region covering the bottom portion only (equivalent in size to the F region) (red). We can see that a much larger portion of the overall ions are being shifted in the PF_6 peak, than in the F peak. This disparity is the first

real answer to our question of why the PF_6 median shift data does a better job in alignment. If there are more PF_6 ions coming from the region which is being shifted the most, these median shifts are doing a better job of characterizing that shift. The ion which best characterizes the regions of greatest shift will do the best job in aligning the other ions peaks.

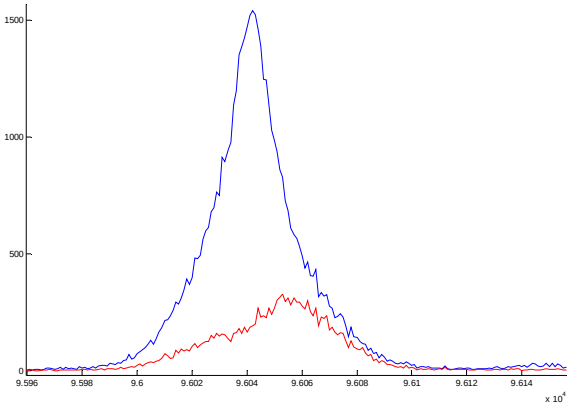

Figure 39a: Comparison of F ions which are shifted (red) to all ions in top drop (blue)

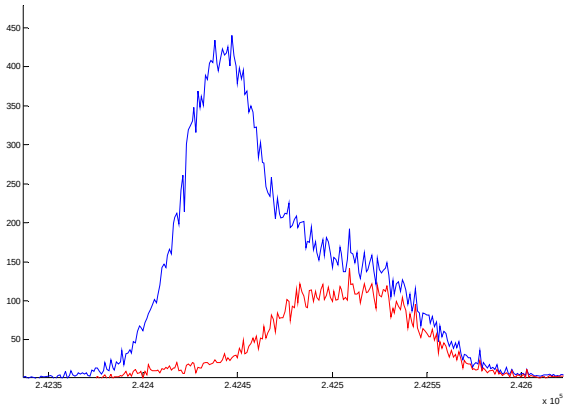

Figure 39b: Comparison of PF_6 ions which are shifted (red) to all ions in top drop (blue)

e. Overall Alignment of F Peak Using PF_6 Median Shifts

Now that we have determined that the PF_6 median shifts are superior to the F median shifts in aligning other ions in the droplets, and we also have an understanding of why the PF_6 data does a better job, the final step is to show the actual improvement that is seen when the F detections are completely aligned using the PF_6 data.

Figure 31 shows the improvement we saw in the top drop based on the PF_6 data. To finish the alignment I first had to align the second drop, based on its spatially dependent shifts of the PF_6 peak (scaled down). Then, I had to align the two drops, again using the scaled shift between the two regions detected by the PF_6 plot. Figure 40a shows the ultimate alignment of the F peak, and Figure 41b characterizes the improvement by finding the decrease in peak width.

By using the PF_6 shift data we were able to improve the resolution of the F peak, a peak which the high shifts affect significantly less that the PF_6 peak, by a factor of two.

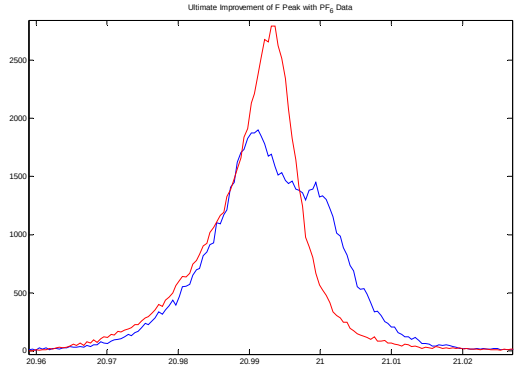

Figure 40a: Comparison of Aligned (red) and Unaligned (blue) F peak using PF_6 Median Shift to Perform Alignment

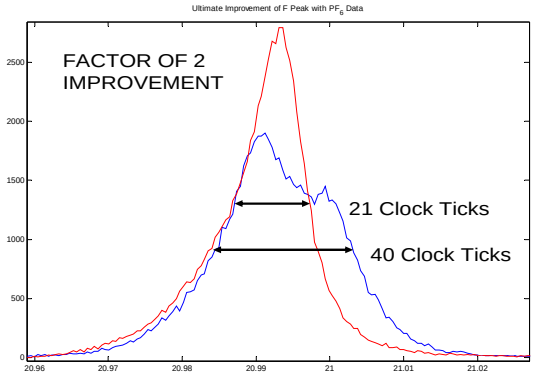

Figure 4ba: Quantification of the Resolution Improvement between Aligned (red) and Unaligned (blue) F peak using PF_6 Median Shift to Perform Alignment

Conclusion

The initial portion of my research was to be able to identify a large number of peaks of different sizes and shapes within a sample spectrum. Once adjusted so it could run with a variety of resolutions the peak picker program was very successful in identifying peaks and revealing their position.

The next stage of the research was to detect and remove global time shifts in the peaks. After using the peak picker to identify the peaks in two spectra from two different regions of a sample, I was able to detect the shifts between the peaks from both regions. It was found that this shift depended linearly on time, and all detections from one region were shifted accordingly to remove the effects of inhomogeneous broadening. This process did result in resolution improvement for the majority of our peaks. In particular, decreases in widths ranging from 10%-25% were seen (figure 18a) with the amplitude increase expected for these resolution improvements.

The next phase of my research was to focus on a sample with more complicated broadening (a sample with two Rtil drops separated by about 25 microns on a silver coated silicon substrate). Initially I divided the region surrounded the top drop into 324 regions and detected the shifts between the medians of the PF_6 peaks in those regions. Once these shifts were calculated, without knowing their cause, I realigned the PF_6 ion data to align its peak. A very significant improvement was seen as the width of the peak was reduced by a factor of five.

I then aligned the second drop through the same process, and found the shift between the aligned top drop and the aligned bottom drop. Through this process I reduced the width of the overall PF_6 peak from the entire sample by a factor of 5, and resolved what initially appeared as two distinct peaks to the a single peak.

Once the spatially dependent PF_6 median shifts were understood, I attempted to use this shift data to align another ion (F). It was seen that the resolution improvement of the F peak using the PF_6 median shift data to align the F detections was comparable to the alignment which results from using the actual F median shift data to align the detections. Unfortunately, the process did not work in reverse, and using the F median shift data to align the PF_6 peak was not as successful as using the PF_6 peak's own median shift data.

To investigate this discrepancy I attempted to identify the cause of the time of flight shifts in the data. It was found that the droplet's geometry alone could not solely account for the time shifts. Instead, it appears the Rtil droplets may shield the ion in its surrounding regions from the electric field. This could account for the time shift by delaying the time it takes for the ions to obtain their final velocity and reach the detector. Due to the presence of the second drop slightly down and the right of the top drop, we would expect to see an increased time shift on the bottom right portion of the top drop due to the increased shielding effect, and in fact, we do see

this increased shift. This hypothesis was further supported when it was found that the shift experienced by the bottom half of the top droplet was equivalent to that experienced by the top portion of the bottom droplet (the only portion of the droplet in the sample).

While we did see the shielding effect in both the PF_6 and the F data, as we would expect, it was also found that a significantly higher number of PF_6 ions which were ultimately detected by the apparatus were experiencing the shift (i.e. they were coming from the bottom portion of the top drop) than were F ions. This would indicate that the PF_6 ions were doing a better job of characterizing the larger shifts in the region, resulting in their superior alignment of the other ions in the region. This superior characterization of the high shift regions is evident in figure 37 where we can see that the PF_6 median shift data is able to align more of the tail of the unaligned F peak then the F median shift data.

The final step to show the ultimate resolution improvement possible with this technique was to align all of the F detections using the shifts detected with the PF_6 ions. After performing this alignment we saw an improvement of the resolution of the F peak by a factor of two. This significant improvement proves the utility of the using a single ion's shifts to align the other ions within the same region.

Ultimately, it was found that it is indeed possible to reduce the effects of inhomogeneous broadening in the TOF-SIMS data and in doing so, improve the resolution of the data. Calculating and removing the time shifts in the data can be done globally to improve resolution or specifically to one ion to improve its resolution. Furthermore, with some investigation into the appearance of the median shifts of a single ion, it is possible to deduce the cause of the shifts, and in doing so identify one ion which characterizes these shifts well. The time shift data from this ion can then be used to align other ions in the region. This will be particularly useful when

the ion's identity is unknown and identification based on its peak shape and position is necessary.

APPENDIX A

1. Serror.m

```
function ERROR = Serror(data,p)

%-----------------------------------------------------
% function ERROR = Serror(data,p)
% Determines the individual error of each peak of a spectrum which the peak
% picker has run.
%
% INPUT:
% data = The spectrum which was analyzed by the peak picker
% p    = structure provided by peakpicker program
%
% OUTPUT:
% ERROR = an 5 dimensional array which provides the position of each peak
% along with its Total Error, Poisson Noise Value, TotalError-Poission
% Noise and Average value of the fitted curve
%

 % creating zero matrix with dimensions to hold error of each peak
len = length(p.PeakPosition);
ERROR = zeros(len,5);
countererror = 0;

% loop to calculate error for each position
for k = 1:len
   % finding indeces in data file matching actual values to fitted
   timeindexpeak = find(data(:,1) == p.RoundUpPeakPosition(k),1);
   timeindexbegin = timeindexpeak - p.left(k);
   timeindexend = timeindexpeak + p.right(k);

   % calculating the average value of the fitted curve
   Sum = sum(p.FittedCurve{k}(:,2));
   Average = Sum / (length(p.FittedCurve{k}));

   % calculating average value of difference between actual and fitted
   Diff = data(timeindexbegin:timeindexend,3)-p.FittedCurve{k}(:,2);
   DiffSq = Diff.*Diff;
   SumDiffSq = sum(DiffSq);
   AveDiffSq = SumDiffSq / length(p.FittedCurve{k});

   % calculating Poisson Error
   Poisson = 1/(realsqrt(Average));

   % calculating Difference between actual Error and Poisson Error
   TotError = ((realsqrt(AveDiffSq))/Average) - Poisson;
   % populating error matrix
   ERROR(k,:) = [data(timeindexpeak,1), (realsqrt(AveDiffSq))/Average, Poisson, TotError,
                 Average];
    end
 return;
```

2. Sfwhm.m

```
function FWHM = Sfwhm(data,p)

%--------------------------------------------------
% function FWHM = Sfwhm(data,p)
% determines full width half maximums of data peaks
%
% INPUT:
% data = actual data
%
% p = structure obtained from running peak picker program
%       - to obtain windows around peaks
%
% OUTPUT:
% FWHM- an array including the full width half maximums of each peak
%
% Note: Must have run peak picker with set values of t_0 and resolution
%       - these values must identify all peak, and have the actual maximums
%         of the peaks falling within the window created by the peak
%         picker.

% creating zero matrix with dimensions to hold fwhm and points where data
% pass half maximum points for each peak
len = length(p.PeakPosition);
FWHM = zeros(len,2);
Cross = zeros(len,100);

% loop to calculate fwhm for each position
for k = 1:len;
   % finding indeces in data file matching actual values to fitted
   diff = p.FittedCurve{k}(end,1) - p.FittedCurve{k}(1,1);
   timeindexbegin = find(data(:,1) < p.FittedCurve{k}(1,1) - diff, 1, 'last');
   timeindexend = find(data(:,1) > p.FittedCurve{k}(end,1) + diff, 1 , 'first');
   % finding what actual maximum and half maxmimum are
   MAX = max(data(timeindexbegin:timeindexend,3));
   HALFMAX = MAX/2;
   % loop to find all points where full width half maximum is pass in
   % data, i acts as place holder for how many cross overs there are
   i=1;
   for j = timeindexbegin+1:timeindexend
      % if crosses half max, put into cross array
      if (data(j-1,3) > HALFMAX & data(j,3) < HALFMAX)
         Cross(k,i) = data(j-1,1);
         i = i+1;
      else
         if (data(j-1,3) < HALFMAX & data(j,3) > HALFMAX)
            Cross(k,i) = data(j,1);
            i = i +1;
         end
      end
   end
   % ends loop
   % Calculating Full Width Half Maximum
   if (Cross(k,1) == 0)
```

```
      FWHM(k,1) = -100;
   else
      final = find(Cross(k,:) > 0, 1, 'last');
      FWHM(k,1) = Cross(k,final) - Cross(k,1)+1;
      position = find(data(timeindexbegin:timeindexend,3) == MAX,1) + timeindexbegin -1 ;
      FWHM(k,2) = data(position,1);
   end;
end;
return;
```

3. Sbestpeaks.m

```
function peaks = Sbestpeaks(resolution, data ,threshold)

%------------------------------------------------------
% function peaks = Sbestpeaks(resolution,data)
% Takes an array of resolutions and runs the peak picker
% with these resolutions collecting all the peaks which
% are found.  In the event that two runs of peak picker provide
% the same peak, the peak with the lower error is recorded.
%
% INPUT:
% resolution = a one dimensional arrary with the different resolutions
%            to be tested.
% data       = matrix with count data (must be three dimensional array
%            first dimension is time of detection and
%            third dimension is counts (i.e. asci data output)
% threshold  = the threshold at which the peakpicker is run
%
% OUTPUT:
% peaks      = a structure containing all the peaks found (including
%            position, amplitude, and fitted curve), the resolution
%            which found the given curve, and the error associated with
%            the fitted curve (both total error and error - Poission
%            Noise).
%
% Note: T_0 is always held at 0.

% procedure works best when Resolution entered in increasing order
resolution = sort(resolution);

% First it runs the first resolution and creates the peaks structure with
% the results from the peak picker
y=BatSimsInCV2(data,0,resolution(1));
warning off;
p=GetSimsPeakV4(y,threshold);
ERROR = Serror(data,p);
ERROR = transpose(ERROR);
originalLen = length(p.PeakPosition);
ResArray = ones(1,originalLen);
ResArray = ResArray.*resolution(1);
```

```
peaks.PeakPosition = p.RoundUpPeakPosition;
peaks.PeakAmplitude = p.Amplitude;
peaks.FittedCurve = p.FittedCurve;
peaks.right = p.right;
peaks.left = p.left;
peaks.TotalError = ERROR(2,:);
peaks.SubError = ERROR(4,:);
peaks.Resolution = ResArray;

% Figures out how many resolutions are provided, and begins loop to
% find new peaks and compare them to get best of overlapping peaks
resLen = length(resolution);

for k=2:resLen;
   % Running peak picker with new resolution and obtaining error
   y1=BatSimsInCV2(data,0,resolution(k));
   p1=GetSimsPeakV4(y1,threshold);

   ERROR1 = serror(data,p1);

   % Compares peaks in peaks structure to peaks in p1 structure.  If they
   % overlap, then the peak with the lowest error is recorded in
   % peaks structure.  If there is no overlap, then the peak is recorded.
   % Which resolution obtained the peak is also recorded.
   for i=1:length(peaks.PeakPosition);
      % find the indeces and position of the two peaks created by the new
      % resolution which surround the peak from the peaks structure
      nextRightPeakIndex = find(p1.RoundUpPeakPosition >= peaks.PeakPosition(i), 1);
      nextLeftPeakIndex = find(p1.RoundUpPeakPosition < peaks.PeakPosition(i), 1, 'last');
      if (nextRightPeakIndex)
         nextRightPeakPosition = p1.RoundUpPeakPosition(nextRightPeakIndex);
      else
         nextRightPeakPosition = 1e10;
      end

      if (nextLeftPeakIndex)
         nextLeftPeakPosition = p1.RoundUpPeakPosition(nextLeftPeakIndex);
      else
         nextLeftPeakPosition = -1000;
      end


      % finding the maximum possible position to the left and right of
      % the original peak (assumes that if two peaks are the same attempt
      % at matching a peak then the peak of the new fitted curve will
      % fall within the width of the original fitted curve) (We will
      % also assume only one peak will ever fall within this range)
      MaxRightPeakPosition = peaks.PeakPosition(i) + (1 * peaks.right(i));
      MinLeftPeakPosition = peaks.PeakPosition(i) - (1 * peaks.left(i));

      % Test to make sure two peaks do not fall within acceptable range.
      % IF they do, and error will be recorded as a -1 in all fields
      % which will indicate to the user that an error has occured
      if (nextRightPeakPosition <= MaxRightPeakPosition)
```

```
   if (nextLeftPeakPosition >= MinLeftPeakPosition)
      newpeaks.PeakPosition(i) = -1;
      newpeaks.PeakAmplitude(i) = -1;
      newpeaks.FittedCurve{i} = zeros(0,1);
      newpeaks.right(i) = -1;
      newpeaks.left(i) = -1;
      newpeaks.TotalError(i) = -1;
      newpeaks.SubError(i) = -1;
      newpeaks.Resolution(i) = -1;
   end % end if
end % end if

% tests to see if the next peak to the right is within acceptable
% range.  If it is, errors are compared and better fit is recorded
% in PeakArray
if (nextRightPeakPosition <= MaxRightPeakPosition)
   if (peaks.TotalError(i) <= ERROR1(nextRightPeakIndex,2))
      newpeaks.PeakPosition(i) = peaks.PeakPosition(i);
      newpeaks.PeakAmplitude(i) = peaks.PeakAmplitude(i);
      newpeaks.FittedCurve(i) = peaks.FittedCurve(i);
      newpeaks.right(i) = peaks.right(i);
      newpeaks.left(i) = peaks.left(i);
      newpeaks.TotalError(i) = peaks.TotalError(i);
      newpeaks.SubError(i) = peaks.SubError(i);
      newpeaks.Resolution(i) = peaks.Resolution(i);
   end % end if
   if (peaks.TotalError(i) > ERROR1(nextRightPeakIndex,2));
      newpeaks.PeakPosition(i) = p1.RoundUpPeakPosition(nextRightPeakIndex);
      newpeaks.PeakAmplitude(i) = p1.Amplitude(nextRightPeakIndex);
      newpeaks.FittedCurve(i) = p1.FittedCurve(nextRightPeakIndex);
      newpeaks.right(i) = p1.right(nextRightPeakIndex);
      newpeaks.left(i) = p1.left(nextRightPeakIndex);
      newpeaks.TotalError(i) = ERROR1(nextRightPeakIndex,2);
      newpeaks.SubError(i) = ERROR1(nextRightPeakIndex,4);
      newpeaks.Resolution(i) = resolution(k);
   end % end if

   % Whether the new peak was used or not, it is now set its
   % amplitude is now set to zero in p1 to prevent it from being counted
   % again when we run through the peak list in p1.
   p1.Amplitude(nextRightPeakIndex) = 0;

end % end if

if (nextLeftPeakPosition >= MinLeftPeakPosition)
   if (peaks.TotalError(i) <= ERROR1(nextLeftPeakIndex,2))
      newpeaks.PeakPosition(i) = peaks.PeakPosition(i);
      newpeaks.PeakAmplitude(i) = peaks.PeakAmplitude(i);
      newpeaks.FittedCurve(i) = peaks.FittedCurve(i);
      newpeaks.right(i) = peaks.right(i);
      newpeaks.left(i) = peaks.left(i);
      newpeaks.TotalError(i) = peaks.TotalError(i);
      newpeaks.SubError(i) = peaks.SubError(i);
      newpeaks.Resolution(i) = peaks.Resolution(i);
   end % end if
```

```
      if (peaks.TotalError(i) > ERROR1(nextLeftPeakIndex,2));
         newpeaks.PeakPosition(i) = p1.RoundUpPeakPosition(nextLeftPeakIndex);
         newpeaks.PeakAmplitude(i) = p1.Amplitude(nextLeftPeakIndex);
         newpeaks.FittedCurve(i) = p1.FittedCurve(nextLeftPeakIndex);
         newpeaks.right(i) = p1.right(nextLeftPeakIndex);
         newpeaks.left(i) = p1.left(nextLeftPeakIndex);
         newpeaks.TotalError(i) = ERROR1(nextLeftPeakIndex,2);
         newpeaks.SubError(i) = ERROR1(nextLeftPeakIndex,4);
         newpeaks.Resolution(i) = resolution(k);
      end %end if

      % Whether the new peak was used or not, it is now set its
      % amplitude is now set to zero in p1 to prevent it from being counted
      % again when we run through the peak list in p1.
      p1.Amplitude(nextLeftPeakIndex) = 0;

   end % end if

   if(nextLeftPeakPosition < MinLeftPeakPosition)
      if (nextRightPeakPosition > MaxRightPeakPosition)
         newpeaks.PeakPosition(i) = peaks.PeakPosition(i);
         newpeaks.PeakAmplitude(i) = peaks.PeakAmplitude(i);
         newpeaks.FittedCurve(i) = peaks.FittedCurve(i);
         newpeaks.right(i) = peaks.right(i);
         newpeaks.left(i) = peaks.left(i);
         newpeaks.TotalError(i) = peaks.TotalError(i);
         newpeaks.SubError(i) = peaks.SubError(i);
         newpeaks.Resolution(i) = peaks.Resolution(i);
      end % end if
   end % end if

end % ending loop which is testing original peak list

% Now we will run a second loop to go through the second peak list of
% the p1 structure.  All peaks which have already considered have amplitudes which are equal
% to zero, so only peaks with non-zero amplitudes will be considered.
% We must keep out index counting the same, so index m is created will
% will continue our counting in the newpeaks structure
m = length(newpeaks.PeakPosition) + 1;
for l=1:length(p1.PeakPosition);
   % Testing if amplitude is non-zero, if it is, information is added
   % to newpeaks structure and m is incremented, otherwise, nothing
   % happens.
   if (p1.Amplitude(l) ~= 0)
      newpeaks.PeakPosition(m) = p1.RoundUpPeakPosition(l);
      newpeaks.PeakAmplitude(m) = p1.Amplitude(l);
      newpeaks.FittedCurve(m) = p1.FittedCurve(l);
      newpeaks.right(m) = p1.right(l);
      newpeaks.left(m) = p1.left(l);
      newpeaks.TotalError(m) = ERROR1(l,2);
      newpeaks.SubError(m) = ERROR1(l,4);
      newpeaks.Resolution(m) = resolution(k);
      m = m + 1;
   end % end if
end % end loop which tests p1 peak list
```

```
   % finally, we set peaks to newpeaks to produce our final result
   peaks = newpeaks;
end % end loop which tests new Resolution
return;
```

4. Sremovebad.m

```
function newpeakstruc = Sremovebad2(peakstruc)

%-----------------------------------------------------
% function newspec = Sremovebad(data)
% Takes a peaks structure and removes all bad peaks which are found in the
% tails of larger peaks
%
% INPUT:
% peakstruc     = the structure with the peak information to be examined
%
% OUTPUT:
% newpeakstruc  = the structure with the new peak information

% position of peaks must be in order
peakpositions = sort(peakstruc.PeakPosition);

indeces = find(peakstruc.PeakAmplitude == 0);
peakstruc.PeakPosition(indeces) = [];
peakstruc.PeakAmplitude(indeces) = [];
peakstruc.right(indeces) = [];
peakstruc.left(indeces) = [];
peakstruc.TotalError(indeces) = [];
peakstruc.SubError(indeces) = [];
peakstruc.Resolution(indeces) = [];
peakstruc.FittedCurve(indeces) = [];
for i = 1 : (length(peakstruc.PeakPosition) - 1);

   % i is the incrementing index of our sorted list, j will be the
   % corresponding actual index in the structure and k is the index of the
   % next peak
   m = length(peakstruc.PeakPosition) - i;
   j = find(peakstruc.PeakPosition == peakpositions(m),1);
   k = find(peakstruc.PeakPosition == peakpositions(m + 1),1);
   if (peakstruc.PeakAmplitude(j) ~= -10)
      if (peakstruc.PeakPosition(k) <= (peakstruc.PeakPosition(j) + (peakstruc.right(j) * 5)))
         if (max(peakstruc.FittedCurve{k}(:,2)) <= max(peakstruc.FittedCurve{j}(:,2)) / 2)
            peakstruc.PeakAmplitude(k) = -10;
         end
      end
   end

end
indeces2 = find(peakstruc.PeakAmplitude == -10);
peakstruc.PeakPosition(indeces2) = [];
```

```
peakstruc.PeakAmplitude(indeces2) = [];
peakstruc.right(indeces2) = [];
peakstruc.left(indeces2) = [];
peakstruc.TotalError(indeces2) = [];
peakstruc.SubError(indeces2) = [];
peakstruc.Resolution(indeces2) = [];
peakstruc.FittedCurve(indeces2) = [];
newpeakstruc = peakstruc;
return;
```

5. Sremovebad2.m

```
function [newpeakstruc,r] = Sremovebad2(peakstruc)

%--------------------------------------------------
% function newspec = Sremovebad(data)
% Runs Sremovebad as many times as necessary to get a final list
%
% INPUT:
% peakstruc     = the structure with the peak information to be examined
%
% OUTPUT:
% newpeakstruc  = the structure with the new peak information

len = length(peakstruc.PeakPosition);
newpeakstruc = Sremovebad(peakstruc);
i = 0;
while (length(newpeakstruc.PeakPosition) ~= len);
   i = i +1;
   len = length(newpeakstruc.PeakPosition);
   newpeakstruc = Sremovebad(newpeakstruc);
end
indeces = find(newpeakstruc.PeakAmplitude == 0);
newpeakstruc.PeakPosition(indeces) = [];
newpeakstruc.PeakAmplitude(indeces) = [];
newpeakstruc.FittedCurve(indeces) = [];
newpeakstruc.right(indeces) = [];
newpeakstruc.left(indeces) = [];
newpeakstruc.SubError(indeces) = [];
newpeakstruc.TotalError(indeces) = [];
newpeakstruc.Resolution(indeces) = [];
return;
```

6. Sshift.m

```
function shift = Sshift(peak1, peak2)

%--------------------------------------------------
% function shift = Sshift(peak1, peak2)
% Takes two peak structures and produces an array which contains the time
% position of the peaks in peak1 and the corresponding shift between that
% peak in peak1 and peak2.  If no correspond peak in peak2 exists, then no
% element is stored in the array.
%
% INPUT:
% peak1 = the peak structure with the peaks from which the shift will be
%         measured
% peak2 = the peak structure whose shift from peak1 will be measured
%
% OUTPUT:
% shift = an array which contains the time position of the corresponding
%         peaks (from peak1) and the shift from peak1 to peak2 for those
%         peaks.
%
% Note: the shift provided will make peak2 match up with peak1, by using
% the opposite, we get the opposite shift

len = length(peak1.PeakPosition);
shift = zeros(len,2);
peak1Position = sort(peak1.PeakPosition);
peak2Position = sort(peak2.PeakPosition);

for i = 1 : len
   % the actual index of the peak being examined in peak1
   j = find(peak1.PeakPosition == peak1Position(i));
   NextSortRightIndex = find(peak2Position >= peak1Position(i),1);
   if(NextSortRightIndex)
      NextRightIndex = find(peak2.PeakPosition == peak2Position(NextSortRightIndex));
      NextRightPosition = peak2.PeakPosition(NextRightIndex);
   else
      NextRightPosition = 1e10;
   end
   NextSortLeftIndex = find(peak2Position < peak1Position(i),1,'last');
   if(NextSortLeftIndex)
      NextLeftIndex = find(peak2.PeakPosition == peak2Position(NextSortLeftIndex));
      NextLeftPosition = peak2.PeakPosition(NextLeftIndex);
   else
      NextLeftPosition = -1000;
   end
      if (NextLeftPosition >= (peak1.PeakPosition(j) - (3 * peak1.left(j))))
      shift(i,1) = (peak1.PeakPosition(j) - NextLeftPosition);
      shift(i,2) = peak1.PeakPosition(j);
   end
   if (NextRightPosition <= (peak1.PeakPosition(j) + (3 * peak1.right(j))))
      shift(i,1) = (peak1.PeakPosition(j) - NextRightPosition);
      shift(i,2) = peak1.PeakPosition(j);
   end
   if (NextRightPosition <= (peak1.PeakPosition(j) + (3 * peak1.right(j))))
```

```
        if (NextLeftPosition >= (peak1.PeakPosition(j) - (3 * peak1.left(j))))
            shift(i,1) = 0;
            shift(i,2) = 0;
        end
    end
end
indeces = find(shift(:,1) == 0);
shift(indeces,:) = [];
return;
```

7. Salign.m

```
function [aregion2,aregiontotal,regiontotal] = Salign(shift,region1,region2)

%-------------------------------------------------------
% function [aregion1,aregiontotal] = Salign(shift,region1,regio2)
% Takes the calculated shift matrix from Sshift, and shifting region2
% according to make it align with region1.  Then it gives the new aligned
% region2, and the combined aligned region of region1 and the aligned
% region2.
%
% INPUT:
% shift   = The shift array between region1 and region2 calculated with
%           Sshift
% region1 = the data structure for region1
% region2 = the data structure for region2
%
% OUTPUT:
% aregion2     = the new data structure for region2 with aligned data
% aregiontotal = the new data structure combining region1 and aregion2
% regiontotal  = the data structure combining region1 and region2
%

line = polyfit(shift(:,2),shift(:,1),1);
SHIFT = round(region2.eventT .* line(1) + line(2));

aregion2.eventT = region2.eventT + SHIFT;
[spectrum,tv] = bspec(aregion2.eventT);
mv = bmcal(tv,2.7542,4.8158,+ 0.211);
aregion2.spec = transpose([tv;mv;spectrum]);

aregiontotal.eventT = [region1.eventT, aregion2.eventT];
[spectrum,tv] = bspec(aregiontotal.eventT);
mv = bmcal(tv,2.7542,4.8158,+ 0.211);
aregiontotal.spec = transpose([tv;mv;spectrum]);

regiontotal.eventT = [region1.eventT, region2.eventT];
[spectrum,tv] = bspec(regiontotal.eventT);
mv = bmcal(tv,2.7542,4.8158,+ 0.211);
regiontotal.spec = transpose([tv;mv;spectrum]);
return;
```

8. SspecChange.m

```
function SpecChange = SspecChange(raw, limits, size)
%-----------------------------------------------------
% function SHIFTCHANGE = SshiftChange(raw, center, size)
%
% Takes a center region from an overall image and find the shift of between
% different regions of the total spectrum and that center region.  How many
% different regions are considered is determined by size.  There are size
% squared regions
%
% INPUT:
% raw       = the data structure of the entire image
% size      = the number of regions the entire image is to be divided into
% limits    = an array with the limits of the region to be divided:
%             limits = [xmin,xmax,ymin,ymax]
%
%
% OUTPUT:
% SpecChange = a structure which contains the new spectrum of each region
%              along with the xmin, xmax, ymin, and ymax of the regions
%

sizeincreasex = (limits(2) - limits(1)) / size;
sizeincreasey = (limits(4) - limits(3)) / size;

% The placeholder in the structure to be created
m = 1;
for i= 0:size - 1; % represents x axis
   for j = 0:size -1; % represents y axis
      xmin = (limits(1) - 1) + floor(i * sizeincreasex) + 1;
      xmax = (limits(1) - 1) + floor((i+1) * sizeincreasex);
      ymin = (limits(3) - 1) + floor(j * sizeincreasey) +1;
      ymax = (limits(3) - 1) + floor((j+1) * sizeincreasey);
      if(j == size - 1)
         ymax = ymax + 1;
      end
      if(i == size -1)
         xmax = xmax + 1;
      end
      region = reducespec(raw,xmin,xmax,ymin,ymax);
      SpecChange.spec{m} = region.spec;
      SpecChange.limits{m} = [xmin,xmax,ymin,ymax];
      m = m + 1;
   end
end
return;
```

9. Smedian.m

```
function median = smedian(specstruc, window)

%--------------------------------------------------
% median = smedian(spec, window)
%
% Given a window around a given portion of a spectrum, median calculates
% the point in that region where half the counts are on one side, and half
% are on the other
%
% INPUT:
% specstruc:   structure created be SspecChange procedure
% window:      the window of the spectrum of which the median is to be
%              calculated (window = [tmin,tmax])
%
% OUTPUT:
% median: a structure containing the median of each region, and the
%         limits of the region
%

for i = 1:length(specstruc.spec);
   index1 = find(specstruc.spec{i}(:,1) <= window(1),1,'last');
   index2 = find(specstruc.spec{i}(:,1) >= window(2),1);
   if(index1)
   else
      index1 = 1;
   end
   if(index2)
   else
      index2 = length(specstruc.spec{i}(:,1));
   end
   Sum = sum(specstruc.spec{i}(index1:index2,3));
   Half = Sum/2;
   j = floor(length(specstruc.spec{i}(index1:index2,1))/4);
   indicator = true;
   while(indicator)
      if(sum(specstruc.spec{i}(index1:j,3)) >= Half)
         m=j;
         indicator = false;
      end
      j = j + 1;
   end
   median.median(i) = specstruc.spec{i}(m,1);
   median.limits(i) = specstruc.limits(i);
end
```

10. Smed.m

```
function MED1 = Smed(median1)
%-----------------------------------------------------
% MED1 = Smed(median1)
% This Program takes a list of median and the limits of the regions to
% which those median correspond and calculates the shift between each
% median and the median which the earliest time.  The program then creates
% a 256x256 matrix into which the individual median shift for each region
% is recorded.  If a particular part of the matrix does not correspond to a
% region with a median shift, it is left at 0.
%
% INPUT:
% median: a structure containing the median of each region, and the
%         limits of the region
%
% OUTPUT:
% MED1: A Matrix describing the median shifts at different points in the
%       image (the 256x256 matrix of Smed correspond to the image produce
%       of a sample)
%

MED1 = zeros(256,256);
sorted = sort(median1.median);
peakposition1 = sorted(1);
for i = 1:length(median1.median);
   diff1 =peakposition1 - median1.median(i);
   MED1(median1.limits{i}(3):median1.limits{i}(4),median1.limits{i}(1):median1.limits{i}(2)) = diff1;
end
return;
```

11. Smedalign.m

```
function [aligned,unaligned] = Smedalign(raw, median, window, factor)

%-----------------------------------------------------
% function [aligned,unaligned] = Smedalign(raw, median, window, factor)
% This program shifts the detections which fall within window by the median
% shift which correspond to the region from which that detection came.  The
% program then calculates various spectrum using this new information
%
% INPUT:
% raw:    the raw data structure created when the raw data is converted
%         (can be specific to a region or time window to decrease
%         processing time)
% median: a structure containing the median of each region, and the
%         limits of the region (input the median shift information of ion
%         you are using to align the detections)
% window: The time range of the detection which should be shifted
% factor: The scale factor being the shifts which are inputted and the
%         shifts which are necessary to align the detection of the specific
%         ion being inputted (factor = sqrt(known ion / ion being align))
```

```
%
% OUTPUT:
% aligned:   A structure containing the aligned time detections and the
%            spectrum generated from the aligneded detections
% unaligned: A structure containing the unaligned time detections and the
%            spectrum generated from the unaligneded detections

sorted = sort(median.median);
%peakposition = sorted(end);
peakposition = sorted(floor(length(median.median)/2));

region = reducespec2(raw,window(1),window(2));
region =
reducespec(region,median.limits{1}(1),median.limits{1}(2),median.limits{1}(3),median.limits{1}(4));
unaligned.eventT = [region.eventT];
diff = round((peakposition - median.median(1)) / factor);
aligned.eventT = [region.eventT + diff];

for i = 2:length(median.median);
   region = reducespec2(raw,window(1),window(2));
   region =
reducespec(region,median.limits{i}(1),median.limits{i}(2),median.limits{i}(3),median.limits{i}(4));
   unaligned.eventT = [unaligned.eventT,region.eventT];
   diff = round((peakposition - median.median(i)) / factor);
   aligned.eventT = [aligned.eventT, region.eventT + diff];
end
[spectrum,tv] = bspec(aligned.eventT);
mv = bmcal(tv,2.7542,4.8158,+ 0.211);
aligned.spec = transpose([tv;mv;spectrum]);
[spectrum,tv] = bspec(unaligned.eventT);
mv = bmcal(tv,2.7542,4.8158,+ 0.211);
unaligned.spec = transpose([tv;mv;spectrum]);
return;
```

[Aubriet, 2003] "Effects of sample preparation on ion yield in the study of inorganic salts by s-SIMS", F. Aubriet, C. Poleunis, and P. Betrand, Applied Surface Science, vol 203-204, 180-183, 2003.

[McDonnell, 2003] "Using Matrix Peaks To Map Topography: Increased Mass Resolution and Enhanced Sensitivity in Chemical Imaging," L. McDonnell, et al. Anal. Chem., 75, 4373-4381, 2003.

[Winograd, 2003] "Prospects for Imaging TOFSIMS: From Fundamentals to Biology", N. Winograd, Applied Surface Science, vol 203-204, 13-19, 2003.