Using cross-correlation to correct for lateral drift in single-molecule localisation microscopy

George Corney

Abstract. Single-molecule localisation microscopy is an important tool for the non-invasive study of sub-diffraction systems and crucial for imaging live cells. As a consequence of the imaging process drift of the sample can occur that limits the resolution. An algorithm was developed to correct for lateral drift without the use of fiducial markers. After investigating various approaches, a method using cross-correlation coupled with Fourier Transform upsampling and kernel density estimation was used with varying degrees of success; the algorithm proved effective in correcting datasets where the structure of the specimen was clustered into isolated groups, however, it was less effective when the structure was more diffuse and complex.

1. Introduction

Recent years have seen significant growth in the development of techniques designed to improve imaging of dynamic nanometer-scale systems. Whilst electron microscopy can offer resolutions below 0.2 nm [12], it cannot be used to effectively image living systems; the process must be carried out in a vacuum, the specimens secured in place, and chemically stained with heavy metals such as uranium or lead. Although optical microscopes are non-invasive and can achieve the imaging speed necessary to study living systems, they're limited by diffraction to $\sim 200\,\mathrm{nm}$ [6]. This limit is governed by Abbe's equation which expresses that the minimum resolving distance, d to distinguish two points (under the Rayleigh criterion) as proportional to the wavelength: $d = \lambda/2NA$. NA is the numerical aperture of the microscope.

1.1. Single-Molecule Localisation Microscopy

Whilst it's not possible to accurately determine the positions of two simultaneously emitting points separated by less than $\sim 200\,\mathrm{nm}$ with an optical microscope, it is possible to do so for one. An isolated point-like emitter is observed as a Gaussian-distributed spread centered about the most likely position of the point - known as the point spread function (PSF). By statistically fitting to estimate the center, it can be shown than the precision is dependent fundamentally upon only the number of detected photons. Assuming background noise, pixel size of the CCD and optics quality have negligible impact, δ_{xy}, the localisation precision is given by [8]

$$\delta_{xy} = \frac{\sigma}{\sqrt{N}} \tag{1}$$

where σ is the standard deviation of the points distribution and N is the number of photons detected. Techniques have been developed that encode depth - or *axial* - information into the PSF, the most common of which is to introduce astigmatism.

By densely populating a target with switchable emitters and triggering emission to occur such that points appear isolated at different times, an image of the target can be constructed from the observed points - potentially enabling resolving capacities that by-pass the limits of regular optical microscopy. If the target is static the resolution of this method depends on the emitter density as well as localisation precision or *optical resolution* - Nyquist-Shannon sampling theorem gives the *structural resolution* as [15]

$$\sqrt{(2.35\bar{\delta_{xy}})^2 + (2\bar{d_{NN}})^2} \tag{2}$$

where $\bar{\delta_{xy}}$ is the average localisation precision and $\bar{d_{NN}}$ is the average distance between neighboring points.

As the observations must be taken over a period of time the target often drifts. Whilst this can usually be corrected for using fiducial markers, in cases where markers are not used or the target itself is dynamical (the study of which is a key attraction of single-molecule localisation microscopy), the structural resolving parameters become more complex; imaging speed is not the only additional factor - the complexity of the target coupled with the effectiveness of reconstruction algorithm contribute as well. In practice, present techniques studying mainly static systems can achieve a localisation precision of $\sim$ 20 nm and an imaging speed of 65 Hz which results in a structural resolution of $\sim$ 50 nm [6][7].

1.2. STORM and PALM

Two popular approaches to SMLM (Stochastic optical resolution microscopy) are STORM and PALM (photo-activated localisation microscopy). Both introduced in 2006, these methods use flurophores (chemical compounds that can be triggered to emit light) as emitters. The fluorophores used must be chosen such that the characteristics best suit the target; some allow higher temporal resolution with faster switching, others denser labeling and some may not be compatible with live samples. Although STORM and PALM are largely the same in their fundamental mechanic of super-resolution imaging, they use differing labeling methods. PALM uses a genetic approach where genes are modified to express the desired flurophore proteins, STORM on the other hand uses anti-bodies carrying flurophores to tag the specimen. The consequence is that in PALM, photobleaching causes the flurophores to expire irreversibly, whereas in STORM they can be designed to be switched on and off many times.

2. Method

The challenge was to develop an algorithm to correct for lateral drift in STORM data without the use of fiducial markers. A collection of real and simulated datasets [1] was used alongside Matlab to develop the method.

The effectiveness of an correction algorithm can be improved by making constraining assumptions about the data - in this case: the drift is small (typically amounting to less than 2% of the dimensions of the field of view) and varies with time, all motion is uniform and translational (there is no structural change with time). Since the fluorophores are stochastically emitting in the case of STORM, it's impractical track individual particles, instead, we have to track structural features.

The field of motion tracking algorithms is well studied and there is a large array of popular approaches. Cross-correlation maximization was settled on as it has been shown to be effective in measuring fine rigid translations between images (for example, it's used in 'Efficient subpixel image registration algorithms' [5] and 'Motion magnification' [9]). An important advantage of cross-correlation maximization is that if strong features are present it is quite resistant to noise. Whilst Iterative Closest Point algorithms are often used for similar problems, investigation found that the sparseness and level of noise of the datasets meant they were ineffective.

2.1. Development

Cross-correlation requires the points to be binned spatially into a frequency grid and temporally into frames. At first, attempts were made to use Matlab's inbuilt *xcorr2* function to compare frames, however this approach proved to be flawed; *xcorr2* can only resolve differences of integer pixels, thus in order to precisely register motion, the binning size must be much smaller than the offset. The point data produced from STORM can be relatively sparse and if divided too finely, features can become dissolved and the computed offsets dominated by noise. Linear interpolation in the binning process was used to try and improve this but it had little effect. Fortunately, *Manuel et al.* provide a Matlab implementation of their translation registration algorithm that uses Fourier Transform upsampling to with cross-correlation - detailed in their paper 'Efficient subpixel image registration algorithms' [5]. Initial results using this implementation were encouraging but left space for improvement.

Further assumptions about the structure of the target can be used to improve effectiveness. With the assumption that the broad structure is smooth, the point density can be effectively increased by interpolated from an estimated probability density function (PDF). Inspired by point cloud motion tracking methods developed for the video game and film industry [13], a bivariate kernel density estimation (KDE) method was used to find the PDF (Figure 1) (implementation provided by *Z. Botev* [2]). See Appendix A.1. This method made it easy to isolate and remove background noise by using a threshold point density on the data.

Computing the KDE was a significant speed bottleneck and after experiments determined there was no serious loss in effectiveness, it was replaced with a faster but less accurate method given in *Eilers, P.H.C. and Goeman, J.J* 'Enhancing scaterplots with smoothed densities' [3] (implementation provided by *P. Perkins* [14]).

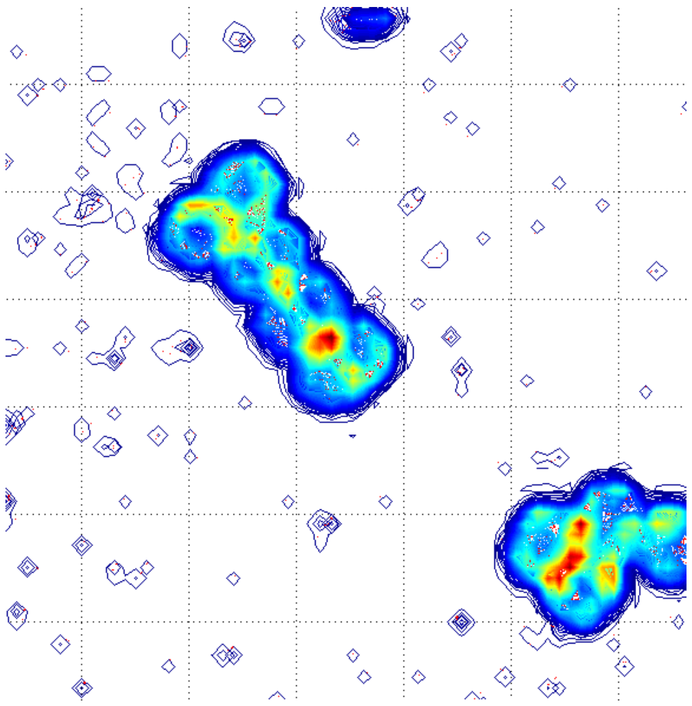

Figure 1: Kernel Density Estimation of point density using a colour scale from blue to red, where blue areas indicate the lowest density and red the highest. The lowest density areas are seen as ringed regions - the algorithm can use these regions to isolate noise.

2.2. Intra-frame motion

A key issue with dividing the data into discrete time frames is that motion that occurs within a frame is unaccounted for - this creates a limit on the final resolution. The solution isn't necessarily to use more frames; shorter time frames means a lower point density per frame and consequently, less precise cross-correlation. It may be possible to reduce or even by-pass this limit if we make assumptions about the motion - for example, if the motion does not vary with time, or if the motion varies smoothly, fitting a curve to the time-offset relationship could be used to approximate intra-frame motion, however this was not taken into account in the presented method.

2.3. Optimisation

The pattern of frame comparison (i.e., the order in which cross-correlation is carried out between frames) has an impact on the algorithms performance and the optimal solution is not clear; is it best to compound pairs of frames and use the result to compare with others? Does compounding frames introduce additional error? After a handful of trials, determining the offset from comparing each frame to the first seemed to produce the best results. The implementation of this method can be seen in Appendix A.2.

The cross-correlation minimization method described above requires a set of input parameters: spacial and temporal bin size, KDE smoothness and noise threshold. The optimal parameters depend on the individual datasets. To estimate them the algorithm is run repeatedly using Matlab's optimization toolbox, each time trying maximizing the result of an image quality analysis test. The test used is a Matlab implementation of *Salvador Gabarda and Gabriel Cristbal's* 'Blind image quality assessment through anisotropy' [4] which promotes sharpness and noiselessness in the final frequency grid.

3. Results

The algorithm was tested with an series of dataset of distinct characteristics using the quality metric described in section 2.3. It was found to be most effective when the target had a mainly clustered or grouped structure, such as the target seen in figure 2 (this result was comparable to using fiducial markers), and least effective when it was diffuse and complex such as figure 3. The more complex structures were also prone to producing occasional wild overestimates of drift and to counter this the kernel density estimation smoothing had to be increased which in turn can reduce the resolution. The effect of different input parameters was investigated and the results of varying KDE smoothing and the number of temporal divisions for the target in figure 2 are given in tables 1 and 2.

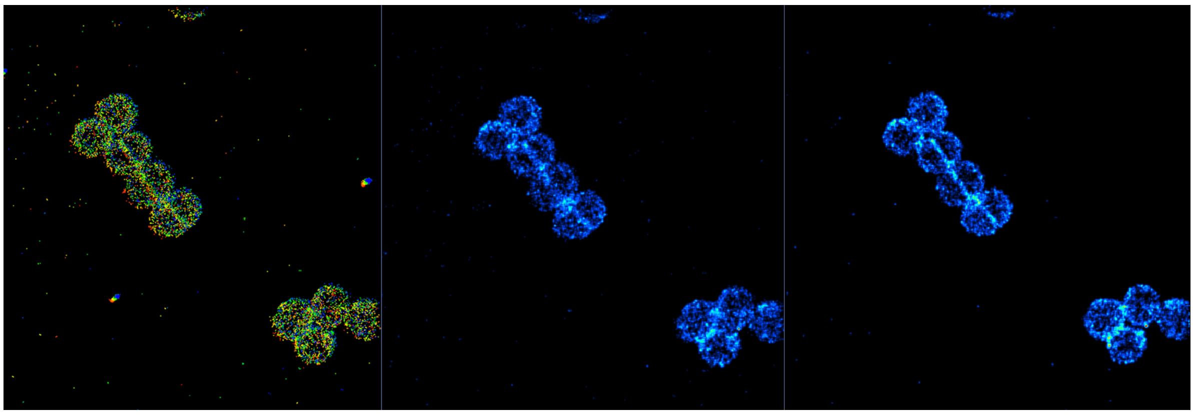

Figure 2: Application of algorithm to real STORM data from supervisor, uncorrected - left & middle and corrected - right. The left image renders the points coloured by observation time; the subtle shift from blue to red indicates the drift. The other two display point density. Dataset contains 9650 points, corrected with parameters: time divisions 5; grid size 256; smoothing 0.5.

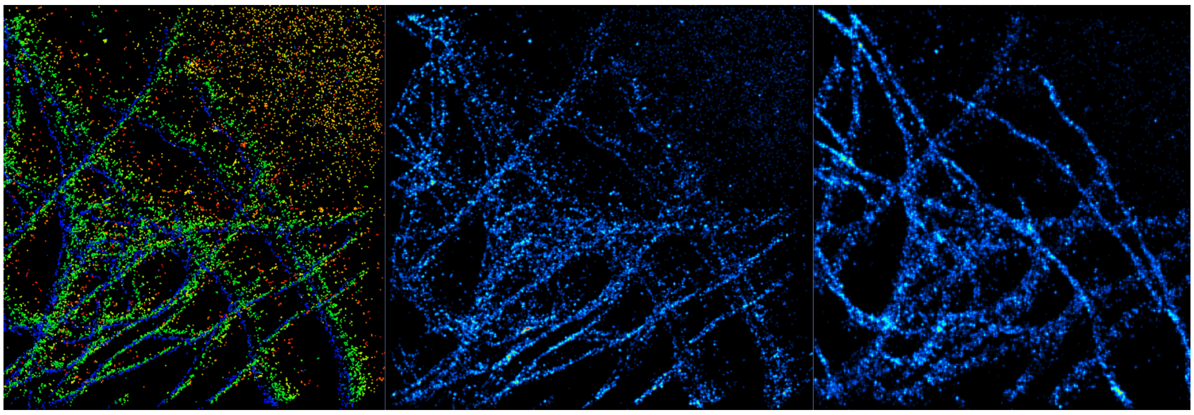

Figure 3: Algorithm less successfully applied to observations of globular protein tubulin taken from [1], with an artificial constant drift. Uncorrected - left & middle and corrected - right. Dataset contains 30831 points, corrected with parameters: time divisions 40; grid size 256; smoothing 4.

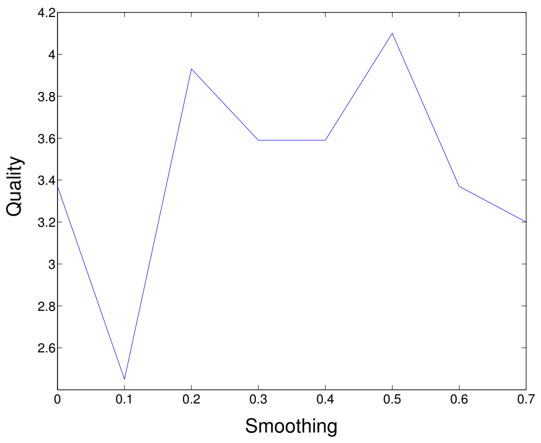

Figure 4: Plot of table 1

KDE Smoothing	Quality Metric
0.0	3.37
0.1	2.45
0.2	3.93
0.3	3.59
0.4	3.59
0.5	4.10
0.6	3.37

Table 1: For the target shown in figure 2: varying the smoothing factor of the kernel density estimation while spatial divisions and noise threshold remain constant at 256 and 0.05 respectively

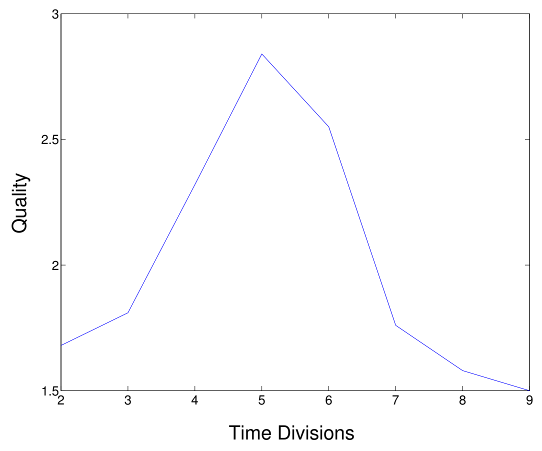

Figure 5: Plot of table 2

Temporal Divisions	Quality Metric
2	1.68
3	1.81
4	2.32
5	2.84
6	2.55
7	1.76
8	1.58

Table 2: For the target shown in figure 2: varying temporal divisions while spatial divisions and noise threshold remain constant at 256 and 0.05 respectively

4. Discussion

Whilst the algorithm was successful in correcting drift some types of structure such as figure 2, it performed poorly with others but still greatly reduced the motion blur effect. Without the use of markers there is a fundamental limit on the corrective abilities of any algorithm that is set primarily by the point density and structural features of the target. In it's current state, however, I believe there is a lot more information that can be utilized before this limit is approached.

4.1. limitations

The method can only correct translational motion and thus any rotational motion presents a limit on the resolution. It is possible to extend the current technique to include rotation but it comes with a large cost on computational complexity. The method cannot be easily extended to handle intra-structural motion - this would require an entirely different approach such as trying to identify and track individual features. However, it can be used to track and correct the broad-scale motion, perhaps before another algorithm is used to analyse more complex motion.

4.2. Further Work

As mentioned in section 2.2, there is lots of work to be done in improving issues with intra-frame motion. The development of a statistical model of the expected motion could constrain assumptions and enable good time-offset curve fitting to reduce this limit, such developments will improve performance on constant velocity datasets such as 3. There are also many possible improvements in optimising the pattern frame comparisons, more experiments should be carried out to understand the elements at play between difference schemes.

Single-molecule localisation methods offer information about the point spread function of each point, this information can be used as weighting to enhance the accuracy of the estimation of the probability density function. Additionally, depth information is usually available with STORM data, since there are no barriers to the extension of the method to 3 dimensions, this data could be easily utilised.

Other approaches could be used in conjunction with this method, such as identifying 'beads' of fluorophores - unusually highly dense collections - and tracking them as particles. In datasets where the signal to noise is high, it might be possible to reconstruct the geometry of the target and use improve correction.

As the algorithm is run repeatedly to determine optimal input parameters, runtime is important, especially for large point counts. Fundamentally, the computationally intensive elements of this technique are parallelizable and thus could be (and would be better suited) executed on a GPU [10][11].

5. Conclusion

Single-molecule localisation methods are fast becoming a powerful method for sub-diffraction limit microscopy and an essential tool in the study of living cells. As the method is dependent on computation reconstruction to achieve the best resolution, effective and advanced algorithms are crucial. To this end, methods were explored and an algorithm developed that used cross-correlation coupled with Fourier Transform upsampling and kernel density estimation to correct for lateral drift without markers in localisation microscopy datasets.

The algorithm proved capable of producing results comparable to using fiducial markers in cases where the points were well clustered but was less effective in cases where they were not. The results suggest there is much room for improvement, especially in combating intra-frame motion and utilising more information available, such as the details of the PSFs.

6. Acknowledgment

Datasets used in this work were provided by supervisor Ashley Cadby of the Electronic and Photonic Molecular Materials Group and the Biomedical Imaging Group, Ecole Polytechnique Fdrale de Lausanne (EPFL)[1]

References

[1] Ecole Polytechnique Fdrale de Lausanne (EPFL) Biomedical Imaging Group. Collection of reference datasets. `http://bigwww.epfl.ch/smlm/datasets/index.html`, August 2013. [online] accessed 20-12-2013.

[2] Z. Botev. kernel density 17204-kernel-density-estimation. `http://www.mathworks.co.uk/matlabcentral/fileexchange/17204-kernel-density-estimation`, October 2009. [online] accessed 20-12-2013.

[3] PHC Eilers and J J Goeman. Enhancing scatterplots with smoothed densities. *Bioinformatics*, 20(5):623–U82, 2004.

[4] S Gabarda and G Cristóbal. Blind image quality assessment through anisotropy. *JOSA A*, 2007.

[5] Manuel Guizar-Sicairos, Samuel T Thurman, and James R Fienup. Efficient subpixel image registration algorithms. *Optics Letters*, 33(2):156–158, 2008.

[6] Sébastien Herbert, Helena Soares, Christophe Zimmer, and Ricardo Henriques. Single-Molecule Localization Super-Resolution Microscopy: Deeper and Faster. *Microscopy and Microanalysis*, 18(06):1419–1429, October 2012.

[7] Rainer Kaufmann, Joerg Piontek, Frederik Gruell, Manfred Kirchgessner, Jan Rossa, Hartwig Wolburg, Ingolf E Blasig, and Christoph Cremer. Visualization and Quantitative Analysis of Reconstituted Tight Junctions Using Localization Microscopy. *Plos One*, 7(2), 2012.

[8] D R Larson, R Thompson, and W W Webb. Precise nanometer localization analysis for individual fluorescent probes. *Biophysical Journal*, 82(1):45A–45A, January 2002.

[9] Ce Liu, Antonio Torralba, William T Freeman, Frédo Durand, Edward H Adelson, Ce Liu, Antonio Torralba, William T Freeman, Frédo Durand, and Edward H Adelson. Motion magnification. *ACM Transactions on Graphics (TOG)*, 24(3):519–526, July 2005.

[10] Yunhui Liu, Qi Zou, and Siwei Luo. GPU Accelerated Fourier Cross Correlation Computation and Its Application in Template Matching. In *High Performance Networking, Computing, and Communication Systems*, volume 163, pages 484–491. 2011.

[11] P D Michailidis and K G Margaritis. Accelerating Kernel Density Estimation on the GPU Using the CUDA Framework. *Applied Mathematical Sciences*, 2013.

[12] D A Muller and J Grazul. Optimizing the environment for sub-0.2 nm scanning transmission electron microscopy. *Journal of Electron Microscopy*, 50(3):219–226, 2001.

[13] Hans-Peter Seidel Oliver Schall, Alexander Belyaev. Robust Filtering of Noisy Scattered Point Data. In *SPBG'05 Proceedings of the Second Eurographics / IEEE VGTC conference on Point-Based Graphics*, pages 71–77. 2005.

[14] P. Perkins. smoothhist2D. `http://www.mathworks.co.uk/matlabcentral/fileexchange/13352-smoothhist2d`, June 2009. [online] accessed 20-12-2013.

[15] Eric J Rees, Miklos Erdelyi, Dorothea Pinotsi, Alex Knight, Daniel Metcalf, and Clemens F Kaminski. Blind assessment of localisation microscope image resolution. *Optical Nanoscopy*, 1(1):1–1, December 2012.

Appendix A. MATLAB Code

Appendix A.1. compareSetsKDE.m

```
function [displacement] = compareSetsKDE(pointsA, pointsB,
   gridSize, bounds)
%compareSetsKDE finds displacement between two point sets
%   Returns displacement [x y] calculated from
   cross-correlation of a
%   kernel density estimation of the points.

W = max([bounds.width bounds.height]);  %find longest dimension
   of dataset
```

```
alpha = W/gridSize;                          %pixels to point unit
   conversion factor (make sure bounds is constant over all
   frames)


[~, densityA, ~, ~]=kde2d([pointsA.x
   pointsA.y],gridSize,[bounds.minX bounds.minY],[bounds.maxX
   bounds.maxY]);
[~, densityB, ~, ~]=kde2d([pointsB.x
   pointsB.y],gridSize,[bounds.minX bounds.minY],[bounds.maxX
   bounds.maxY]);


[out, Greg] = dftregistration(fft2(densityB),
   fft2(densityA),1000);%sub-pixel cross-correlation

d.x = out(3)*alpha;
d.y = out(4)*alpha;

displacement = struct2dataset(d);  %push shift in
   point-position units
end
```

Appendix A.2. driftCorrection.m

```
% %Config
% pointsToCorrect = pointsWithoutMarkers;
% nDivisions = 8;
% gridSize = 2^10;
% smoothing = 1;  %only used in 'compareSetsSmooth' mode
% noiseThreshold = 0.00; %if 0, do not remove noise

function [pointsCorrected, quality] = driftCorrection(
   pointsToCorrect, nDivisions, gridSize, smoothing,
   noiseThreshold )
if nargin < 2 nDivisions =   8;    end
if nargin < 3 gridSize =   2^8;   end
if nargin < 4 smoothing =  1;      end
if nargin < 5 noiseThreshold =  0.05;end

bounds = getBounds(pointsToCorrect);

%Noise removal
[~, outliers] = smoothhist2D([pointsToCorrect.x
   pointsToCorrect.y], 2, [gridSize gridSize], [],[],
   noiseThreshold, '');
if noiseThreshold > 0
```

```
    pointsToCorrect = removerows(pointsToCorrect, outliers);
end

%Core vars
pointCount = length(pointsToCorrect.x);

%allocate corrected dataset
pointsCorrected.x = zeros(pointCount, 1);
pointsCorrected.y = zeros(pointCount, 1);
pointsCorrected.t = zeros(pointCount, 1);

divisionWidth = floor(pointCount/nDivisions);

%Pile Method: Compare successive fames to first frame
nPairs = nDivisions-1;
i = nPairs;

%low frame
lf_b = (1-1)*divisionWidth + 1;
lf_t = (1)*divisionWidth;
rangeLow = lf_b:lf_t;

low.x = pointsToCorrect.x(rangeLow);
low.y = pointsToCorrect.y(rangeLow);
low.t = pointsToCorrect.t(rangeLow);
pointsCorrected.x(rangeLow) = low.x;
pointsCorrected.y(rangeLow) = low.y;
pointsCorrected.t(rangeLow) = low.t;
while i>0
    %Find displacement
    %high frame
    hf_b = (i)*divisionWidth+1;%bottom
    hf_t = (i+1)*divisionWidth;%top
    rangeHigh = hf_b:hf_t;

    high.x = pointsToCorrect.x(rangeHigh);
    high.y = pointsToCorrect.y(rangeHigh);
    high.t = pointsToCorrect.t(rangeHigh);

    %displacement = compareSetsKDE(low, high, gridSize, bounds);
    displacement = compareSetsSmooth(low, high, gridSize,
       bounds, smoothing, 0);

    %Displace high by displacement
    pointsCorrected.x(rangeHigh) = high.x - displacement.x*1;
    pointsCorrected.y(rangeHigh) = high.y - displacement.y*1;
    pointsCorrected.t(rangeHigh) = high.t;
    %disp( sprintf( 'Frame %d dispacement %f %f', i,
       displacement.x, displacement.y) );
    %loop
    i=i-1;
end
%end Pile method

%figure(1);
[density, outliers, ax, ay] = smoothhist2D([pointsCorrected.x
   pointsCorrected.y],0.0,[2^9 2^9],[bounds.minX
   bounds.minY],[bounds.maxX bounds.maxY],0,'');
%set(gca,'YDir','normal');
%axis image;
%colormap(jet);

%writeOutXYT(pointsCorrected, 'corrected.xyz');
quality = blindimagequality(density);
%disp(quality);
%q(end+1) = blindimagequality(density)
%displayPoints(pointsCorrected);
```