Spatial and dynamic organization of molecular structures in the cell nucleus

Brouwer, A.K.

CHAPTER 2

STACKS:
a software program for particle tracking
in living cells

STACKS: a software program for particle tracking in living cells

Vrolijk H., Brouwer A.K., Dirks R.W., Tanke H.J.

Department of Molecular Cell Biology, Leiden University Medical Center Leiden, The Netherlands

Introduction

Many processes in the mammalian cell are dynamic. In order to obtain better understanding of these processes the dynamic properties of cellular structures have to be measured. Nowadays the commercially available fluorescence confocal and wide-field microscope workstations offer excellent facilities to visualize the three-dimensional spatial organization of the cell and to study the movement and interaction of nuclear and cytoplasmic particles. However, the software provided by the manufacturers of these workstations is more or less dedicated to the control of the microscope, the acquisition process and the visualization of the acquired 2D and 3D image series in time, while at present the analysis tools are mostly limited to simple measurements. However, dedicated image processing and image analysis software is required for more complex research questions, such as the tracking of moving structures in a cell.

Particle tracking is applied using a variety of biophysical techniques, such as microrheology (Tseng et al., 2002; Weihs et al., 2006; Waigh, 2005), magnetic tweezers (Bausch et al., 1998), optical tweezers (Ashkin, 1997) and is widely used for particle imaging velocimetry (Adrian, 2005; Grant, 1997). A wide range of methods have also been described to track particles in microscopic optical images (Bacher et al., 2004; Miura, 2005; Cheezum et al., 2001; Crocker & Grier, 1996; Sbalzarini & Koumoutsakos, 2005; Carter et al., 2005).

In this paper we describe a software package, called STACKS, that was developed to handle research questions that could not be solved with the standard software presently available on commercial systems, such as the tracking of particles in 2D and 3D time series and the measurement of the dynamic characteristics of these particles. We considered it important that STACKS should be able to operate directly on the data formats as produced by commercial microscope systems without a conversion step for the image data. Furthermore, the program should be user-friendly, provide visual feedback on each operating step and allow sufficient flexibility to cope with different cell types and images of varying quality. The software should also allow a sufficient amount of user interactivity for tuning the analysis procedure and editing the image stack, for instance for the removal of artifacts, which may disturb the analysis, or for correction of partly incorrect image segmentation. In STACKS a number of filter operations are provided to enhance the original image data. As a 4D image stack easily consists of more than 1000 images, these filter operations become time consuming when they are based on normal CPU processing. In STACKS many of these operations have been accelerated by making use of the GPU processor of the video board. In this paper we compare the two approaches with respect to the efficiency of the image operations. To illustrate the capabilities of the STACKS program determination of the dynamics of chromatin during different phases of the cell cycle was chosen as a test model. Chromosomes of eukaryotic cells have been shown to occupy discrete territories during interphase (Manuelidis, 1985; Trask et al., 1988). Transcriptionally competent regions preferentially localize to the periphery of these chromosomal

territories (Verschure et al., 1999), indicating that chromatin is dynamic. This type of organization is observed not only at the level of the chromosome, but also at the level of a single genetic locus in the total space of the nucleus (for a review, see Spector, 2003). Nuclei in higher organisms are known to undergo extensive changes in organization as they progress through the cell cycle and during development (for a review, see Francastel et al., 2000). For example, the brown^{Dominant} (bw^{D}) chromosome of Drosophila melanogaster contains a large block of heterochromatin near the end of chromosome 2 (2R) (Platero et al., 1998). This distal block associates with centric heterochromatin of the second chromosome (2Rh). The association between bw^{D} and 2Rh is not apparent until at least 5 hours into G1 (Csink & Henikoff, 1996; Dernburg et al., 1996). Also, as soon as the nuclear membrane forms in early G1, centromeres have been observed to rapidly disperse throughout the nucleus. Coalescing and dispersing of centromeres in cultured cells was observed in late G_2 and early G_1, respectively (Manuelidis, 1985).

In this study telomeres were chosen as a marker for studying the dynamics of chromatin. Telomeres in living cells were labeled by transfecting U2OS cells with GFP fused telomere binding proteins TRF1 and TRF2. Time-lapse movies were recorded of moving telomeres during different phases of the cell cycle. GFP- tagged proliferating cell nuclear antigen (PCNA) was hereby used as a life cell marker to distinguish the different phases of the cell cycle, and as a counterstain to correct for cell movement. The movements of telomeres were quantitatively analyzed using STACKS and telomeres were found to be significantly more dynamic in the G1 phase than in other phases of the cell cycle (p-value = 0.05).

Material and Methods

Hardware

In our laboratory several fluorescence microscope workstations are present to perform live cell analysis: the AF6000, the SP5 (both Leica Microsystems, Wetzlar, Germany) and the LSM 710 (Carl Zeiss Microimaging, Jena, Germany). The AF6000 is a wide-field system and consists of an inverted DMI 6000B microscope equipped with a metal halide bulb, an automated motorized z-galvo stage for 3D imaging and a climate chamber for live cell imaging. The SP5 is a confocal laser scanning microscope (inverted) system, also equipped with a climate chamber, and the LSM 710 is a multiphoton confocal laser scanning (upright) system equipped with objectives with a large working distance and a special stage for intra-vital microscopy. The Leica SP5 is equipped with an Argon laser for 405-510 nm, a solid state laser and a helium-neon laser for respectively the 561 nm and 633 nm line and a two-photon laser with a spectral range of 710-990 nm. The Leica systems are controlled by the LAS AF software for image acquisition and analysis. The LSM 710 is equipped with similar lasers as the Leica SP5, a two photon laser for the range of 700 – 1060 nm and is controlled by the ZEN software system.

The software program described in this paper, STACKS, can be used on a regular personal computer running Windows XP, Vista, or Windows 7. It was tested on the 32 bit versions of these operating systems. The graphics processing unit (GPU) based image processing functions are currently only supported for Nvidia (Santa Clara, California, USA) video boards from the Geforce 7 series or higher with at least shader level 3. The PC used for image analysis was a Precision 380 system from Dell (Round Rock, Texas, USA) equipped with a GTX8800 Nvidia video display adapter.

Software system

The program STACKS has been developed to cope with research questions, which could not be solved using the software that comes with commercial confocal systems and live cell workstations. It performs dedicated processing and analysis of time lapse 2D and 3D image stacks, although it can also handle single 2D and 3D images. Both 8 and 16 bit grey-value image stacks are supported as well as 24 bit color image stacks.

All image operations in STACKS are operated from the main menu. The source for the operation is always the top window. When additional images or parameters are required for the operation, a dialog is shown to ask the user to specify the additional input and/or output stack. Each image operation results in a new image stack. When the user selects an existing stack as being the result of an operation, it will be overwritten. The user has also control on which part of the image stack an operation is performed. There are 4 modes of operation. It is possible to process 1) the entire image stack, 2) the 3D image for the current time point, 3) the z-slice for all time points or 4) just a single image. By clicking on the corresponding icons in the toolbar the user can specify on which part of a stack an operation is performed.

Image Visualization

An image stack is normally displayed in a re-sizable window, which maintains the original aspect ratio of the image. Two additional scrollbars can be present. The horizontal scrollbar at the top of the window allows the user to select a time point within the stack, whereas the vertical scrollbar at the left allows the selection of a certain z-slice; the traditional scrollbars of

the window are used for zooming. In case of 3D stacks one may also display the horizontal and vertical cross sections of an image stack, as shown in Figure 1. A crosshair is then shown in the original image to indicate where the cross sections are located. By moving the mouse over an image, the coordinates and grey-value information of the corresponding pixels are shown at the status bar at the bottom of the screen and optionally also as a tooltip. It is also possible to display a 3D representation of an image stack. This has been implemented using the public domain package of the visualization toolkit, VTK [http://www.vtk.org/].

Data I/O

STACKS can import experiments of time-lapse 2D and / or 3D image data saved in the Leica LIF and the Zeiss LSM file format and can also export the resulting image sets in both formats, although the specific microscope settings of the original data files will be lost in these new data files. The data sets can be read in again by respectively the LAS software (Leica) or by the ZEN software from Zeiss. Another option in STACKS is to read a set of TIF files from a folder; the names of the TIF files should be of a special format containing time point, z-slice and channel information in order to reconstruct the time-lapse 3D image set from it. The supported filename formats are compatible with older confocal laser scanning systems from Leica and with Tikal, a software environment for the tracking of 3D microscopic particles as described by Bacher (Bacher et al., 2004). Analysis results can be exported as Excel spread sheets.

Management of the stacks

In the program there is currently support for the simultaneous use of 10 4D image stacks and 10 3D image stacks. For each 8 or16 bit grey-value stack there is also an additional 8 bit stack present, which is used for display purposes. This can for instance be obtained after contrast stretching of the 16 bit stack. In this way the original data values stay intact. Additionally the horizontal and vertical cross sections can be calculated and from each 4D stack the maximum projection can be obtained or a z-slice can be selected as being the projection stack. This would imply, that the total amount of memory used by image data would become about 11 Gbytes in case of 10 stacks of 512x512 pixels (16 bit + 8 bit, or 24 bit) with 40 slices and 25 time points including cross sections and projections; evidently too large to handle in memory for a normal PC. Therefore we have chosen for an approach by which only the current visible images and the corresponding images holding the original data of each stack are present in memory. At the time that new image stacks are read from disk or created by image transformations, copies of the complete stacks are created in the /TEMP directory of the system disk which are accessible through fast random access. In this way access to the data is optimized for display purposes, so that scrolling through the stack appears to occur instantaneous to the user. This approach will result in extra overhead when stacks are processed and have to be written back to disk.

Segmentation

In order to segment the particle images from the background various methods are present. Global thresholding based on the image background can be applied for the complete data set; the user may adjust this threshold per individual image, per slice in time or per time-point. Thresholding can be performed on the 8 bit image stack obtained by contrast stretching or directly on the 16 bit original data. However, it is often easier to segment small structures, such as centromeres or centrioles, after performing a 2D top-hat transformation on the image set in order to reduce the nonspecific fluorescence within the cell nucleus or the fluorescence fluctuations in the background. Other image transformations are provided to enhance the images

before the segmentation step, such as contrast enhancement, convolution filters, Fourier filters, median and min-max filter operations.
The 2D or 3D watershed algorithm respresents a more automatic segmentation method, which will detect the particles as being local maxima in the 2D or 3D image sets and will find the borders of their domains as being the watershed lines between the individual particles. Within each domain a local threshold is automatically determined to segment the particles from the background. The advantage of the latter algorithm is that touching particles are often correctly separated. A possible disadvantage is that this operation may lead to over-segmentation.

Segmentation results in a new 8 bit image stack, from which one bit corresponds with the result of the segmentation, namely object or background. Other bits are reserved for segmentations of other image stacks, such as for a second label or for the counterstain. Binary operations between different bit-planes are provided, like logical AND, OR, or EXOR. Also, image transformations can be applied, such as erosion, dilation, and skeleton. The highest bit is reserved for human interaction, as will be discussed in the following paragraph. The user can select which bits of a binary stack will be displayed and edit the binary stack effects in the visible bit-planes.

Correction global cell movement
The program also corrects for global cell movement. The orientation and translation of the cell is calculated for each time-point based on the counterstain image of a 2D image set (or on the maximal projection in case of a 3D image set). Alternatively, one may also select particles which are known to be immobile. The results are then used to perform a translation and / or rotation correction for the original image sets. Obviously, such corrections are essential, for instance the kinetic characteristics of the tracked particles in a cell nucleus are not very meaningful without a correction for global cell movement.

Particle tracking
In order to track particles, such as telomeres and centrioles, at first the user has to segment the 3D or 4D stack. When thresholding is used for segmentation, object labeling is the next step of the process. Object labeling will detect separate objects based on 8-connectivity either in the 2D or 3D images of the binary stack and will create a new stack, called the label stack, where by all pixels of every individual particle obtain a unique index, that corresponds to a pseudo-color. However, when the watershed algorithm is applied for segmentation, both the binary stack and the label stack are directly created during the process.

Following object labeling position, size and total density of each particle are measured for all time-points and the tracks are determined by linking those particles between successive time points, which have the highest probability of being the same objects based on these features. Particles, which have been classified as being the same, are relabeled with the same pseudo-color in the label stack. This allows the user to easily verify by scrolling between the time points in the bale stack, how successful the classification was performed. Figure 1 shows the various stacks involved in the tracking process and the 3D representation of the tracks found. Eventually tracks can be split and reconnected by the user to correct for errors made by the automatic procedure. Finally kinetic parameters such as mean squared displacement are measured to characterize the movement of each individual particle according to the following formula:

$$MSD_p(\Delta) = 1/(N-n) \sum_{m=1}^{N-n} [\, pos((m-1).\tau + \Delta) - (pos((m-1).\tau)\,]^2$$

with $\Delta = n.\tau$ τ – measurement interval and n - an integer
pos - position vector at a certain time point
N - the total number of time points

Thus the mean squared displacement is averaged over all possible time intervals measured. This will inevitably result in a larger standard deviation for increasing values of n, as the number of interval pairs decreases. A similar function, called MSD_t, was also calculated; the position vector in the formula above is then replaced by the distance covered between two time intervals. This function provides additional information about the mobility of the particles over time.

GPU programming

Nowadays, personal computers are equipped with video boards that provide high processing power for use in video games and multimedia applications. These graphics processing units (GPUs) are very efficient in parallel processing of small programs (called shaders) on sets of vertices and pixels, which makes these boards very suitable for general purpose image processing. Many papers have already been published on this subject (Moreland et al., 2003; Strzodka et al., 2003, 2004; Farrugia et al., 2006)

A number of image operations have been implemented in STACKS using the processing power of the GPU. Examples are simple mathematical operations, like adding, subtracting, multiplying and dividing of images. They are included, as they are internally used for more complex operations. Morphological operations have been added, like erosion, dilation, opening, closing and the top-hat transform. These min/max operations are implemented using the so-called “ping-pong” technique. This is caused by the fact that source and result image, which are defined as 2D textures in GPU memory, are not allowed to be the same for a shader. In the first pass the minimum or maximum is determined with the 4 closed neighbours only. In the next pass the result serves as the source while the minimum or maximum is determined with the 4 diagonal neighbours only and this is alternately repeated until the desired size of the neighbourhood is reached. By defining an extra image texture it can be prevented that the source is changed and that the result is obtained in the correct texture at the end of the operation. The access to texture memory is highly optimized by parallel processing, so that these operations are very efficiently handled by the GPU even when many passes are involved.

Also other filter and transformation operations based on the GPU have been implemented in STACKS, like Sobel, Laplace, Canny edge, Gaussian, median, Kuwahara, convolution up to a 40 x 40 neighbourhood, unsharp mask, a distance transform (Rong et al., 2006), nearest neighbour deconvolution and some specific color transformations, such as RGB to HSI and the inverse transform, component blend, component threshold, and color deconvolution (Ruifrok et al., 2001) Furthermore the 2D fast Fourier and inverse fast Fourier is supported including low-pass, band-pass and high-pass Gaussian and Butterworth filters based on the 2D fft. Most of the image operations have been implemented for 8-bit and 16-bit grey-value images and 24-bit color images and 32-bit float and complex images for the Fourier transform. Also 48-bit color images (16-bit for red, green and blue) have been programmed for the image operations on the GPU, but these images are not yet supported by STACKS.

Initially the shaders have been developed using the HLSL (high level shader language, Microsoft) DirectX and Direct3D. More recently Nvidia introduced CUDA as the new high level language for general purpose processing on GPU boards. For STACKS two equivalent libraries were developed, one based on HLSL and one on CUDA. Both languages have their advantages and disadvantages. In HLSL the result of a shader program can again be a 2D texture. Because the texture format is defined apart from the shader program, the same shader program can be applied for different image types such as 8-bit grey, 16-bit grey or 24 bit color. In CUDA only the source images can be 2D textures, while the result image of a shader program must be a 2D array, so that different shader programs are necessary for the different image types. The advantage of using 2D textures as input instead of 2D arrays, is that the structure is defined in the texture. Therefore the boundary conditions will be at the borders of the images and there is no need to test if neighbouring pixels exceed the borders of the source images in the shader programs themselves. CUDA has the advantage that also other memory compartments of the GPU board can be addressed, such as the local memory of the shaders. This can lead to more optimized implementations of the shader programs, as is described for the 2D convolution filter (Podlozhnyuk et al., 2001). Additionally there is a large community using CUDA for general purpose processing and a lot of examples are present in the Nvidia software development kit and on their website, which eases the development using CUDA.

In this paper the different implementations of the image operations in HLSL and CUDA are compared with the normal software implementation of these operations carried out by the CPU. The necessary overhead for processing images by the GPU, such as the additional transfer of images to and from GPU memory and the setup of textures on the GPU board will be taken into account. Image management was added to the GPU libraries in order to minimize the overhead for creating new 2D textures and reserving memory space on the GPU board. In this way new 2D textures are created only when they did not exist yet, but when a 2D texture with the same specifications has already been defined during a previous operation, textures can be reused for different images with the same specifications. Especially for processing 3D and 4D image stacks were all 2D images of the stack are of the same dimensions and the same image type, image management is very useful as new textures will be defined only during the processing of the first image of the stack.

Interactivity

The program STACKS allows a large amount of user interaction on the image stacks. For instance one may separate touching objects or reconnect them, so that mistakes made by the segmentation process can be corrected. Other functions are included to delete objects or regions, or to select objects or regions and then erase the unselected parts. In this way further analysis can be restricted to a part of the image stack or to the objects of interest only. The actual drawing and selection is performed on the images of the binary stack, although the user can draw or point with the mouse in any of the image stacks with the same dimensions as the binary stack. Once an object is selected in the binary stack, there are also options to move, duplicate or erase the corresponding original object in the grey-value stack. This may be useful for cleaning up dirt that has moved in during the acquisition of the images. Depending on the selected mode of operation interactions are carried out on the 2D image only, on the 3D image, on all time-points of a slice, or on the complete 4D stack. User interaction is also possible on the particles after tracking. Tracks can be selected or erased or split at a certain time-point or two tracks can be connected at the end of the tracks found. However, in this case the program is aware how these tracked objects are connected through the slices and over time-points, so that the interactions on tracks are correctly handled in 3D and 4D.

Measurements

The program STACKS was originally developed for the tracking of nuclear particles, but in order to extend its possibilities, the functionality to execute separate measurements on individual particles has been added. The user can select which features have to be measured, like size, density and shape. This can be applied for more than one color if required. STACKS also provides distance measurements between particles and selected objects. For instance, it may be of interest to determine how fast viral particles move through the cell. Measurements can also be performed within regions of interest (ROIs), that are drawn by the user in the form of simple mathematical shapes like squares, circles, rectangles, ellipses and lines, but also as freehand drawn closed regions. Selected objects may be converted into ROIs as well. ROIs have a constant shape, position and size for images of a stack in contrast with segmented objects. However, ROIs can be duplicated and moved through the image, so that exactly the same area's can be measured on different places within the images of a stack allowing for instance FLIP and FRAP measurements to be performed. The results of all measurements are shown in the result window and can be saved on disk as an Excel spreadsheet.

Macro recording

Macro recording was added to STACKS to easily execute a sequence of instructions on different data sets. STACKS supports the use of 10 different macro's simultaneously, which is sufficient for most applications. Some operations, such as thresholding require user interaction. These operations are also recorded in the macro, but when the macro is executed the user can specify, whether he wants to be prompted to perform the interaction again on a new data set, or whether the macro should apply the same settings, as defined during the recording of the macro. Macro's can be executed once or repeatedly. The latter can be very useful especially when the image content is changed in each pass of the execution. For instance when a particular part of the nucleus is photo-activated and the user likes to study how the activated subcellular structures or particles diffuse through the nucleus. The user may select the photo-activated part at the first time-point and measure the intensity within this area as function of time. However, by dilating the region in each pass of the macro this intensity can also be calculated as function of the distance to the original region.

Macro's can be stored to disk in the so called "preferences" file, which contains also all other parameters and settings. How a given data set was analyzed can be saved in this way. The settings also specify other parameters such as predefined positions and sizes of the windows, the default lookup tables for the windows, and the parameters which are used for the analysis, such as the minimum object size for objects to be detected and the maximum distance over which objects still should be considered as being the same between two successive time points during the tracking analysis.

Cell Culture

A study of the dynamics of telomeres during different phases of the cell cycle was performed in order to illustrate the possibilities of STACKS for particle tracking. Human osteosarcoma cells (U2OS) were cultured at 37ºC on 3.5 cm glass-bottom culture dishes (MatTek) in Dulbecco's modified Eagle's medium (DMEM) without phenol red and containing 1.0 mg/ml glucose, 4% FBS, 2 mM glutamine, 100 U/ml penicillin, and 100 µg/ml streptomycin, pH 7.2 (all from Invitrogen).

Plasmids and cell transfection

The coding sequences for TRF1 and TRF2 have been cloned into the DsRedExpress vector (Clontech) according to standard procedures. The GFP-tagged proliferating cell nuclear antigen (PCNA) protein was a gift from M.C. Cardoso. Cells were transiently transfected with 0.5 μg vector DNA using lipofectamine 2000 (Invitrogen).

Live cell imaging

Wide-field fluorescence microscopy was performed on the AF6000 multi-dimensional workstation for live cell imaging. 4D image stacks were collected using a 63×NA 1.25 HCX plan Fluotar objective in combination with the automated motorized z-galvo stage. During imaging, the microscope was heated to 37 ºC in a CO_2 perfused and moisturized chamber. Generally, image stacks were collected every 30 seconds for 10 minutes. Image deconvolution was performed using the Leica software. For each experiment and cell type at least six image series were analyzed.

Results

Disk overhead

As described the program STACKS only has the current visible image of each stack present in memory. This will introduce extra overhead when a complete stack is processed, namely to fetch all images from disk and to restore the resulting images back to disk. An overview of the overhead measured is given in Table 1. The overhead was measured for 3 stacks of 25 time-points and 40 slices with varying image sizes of 256 x 256, 512 x 512 and 1024 x 1024 pixels. The following times were measured: the time to read the grey-value stack for the first time and create the random access files (8-bit for display purposes and 16-bit original data values) in the /TEMP directory, the time to only read the complete stack and the time to read and write the stack back to disk. The latter two measurements give an indication respectively for the overhead when the user scrolls through the stack and when a complete stack is processed. In Table 1 the results are given using two different disks, namely a 1TB disk, type HD103UJ, from Samsung (Seoul, Korea) and the solid state disk of 120 GB, type SSD2 from OCZ (San Jose, California, USA). If the time to read a stack is divided by 1000, the overhead for scrolling from image to image through the stack is obtained. This time is very short, so that scrolling appears to occur instantaneous to the user.

Comparison between CPU and GPU processing

In Table 2 a comparison is shown for the various image operations using respectively shaders written in HLSL and CUDA and the software equivalent of these operations written in C++. The figures are given as the processing time per image. The time to put an image on the GPU board and the time to get an image from the GPU board are part of the processing time. It appears that image processing using the GPU is more effective when image operations become more complex or when kernel sizes increase. The GPU is for all measured image operations in Table 1 much faster than the software equivalent despite the overhead of transferring images between the GPU board and computer memory. The shader programs written in HLSL are almost always faster than those written in CUDA. The difference for processing color images is even more distinct. However, there are possibilities to improve most of the CUDA shader programs written for color images and therefore some improvement is to be expected. On the other hand the CUDA implementation of the functions based on the fast Fourier transform is significantly more efficient that the HLSL equivalent. In Figure 2 the dilatation or MAX operation is shown for a 16-bit image with varying image and kernel sizes. As expected, there is a linear relationship with increasing kernel size. An estimate for the time necessary to transfer an image between the GPU board and computer memory is obtained at the points where the lines cross the y-axis.

Tracking

In this study the dynamics of telomeres during different phases of the cell cycle were measured during mitosis using the STACKS software. Global cell movement was first determined based on the nuclear image stained with GFP-PCNA. The translational and rotational movement from the cell was then removed from the stack containing the telomeres, after which the movement of individual telomeres was tracked. The MSD was measured for 6 cells for each cell cycle phase. Using GFP tagged PCNA we were able to discriminate the G1 and the G2 phase of the cell cycle, and also three different stages with in the S-phase (Leonhardt et al., 2000). In Figure 3 the MSD is determined in two different ways, one method uses the distance that the telomeres have traveled, and is referred to as MSDp, and the other method determines the area in which the telomeres have moved, which is the MSDt. The extreme values were omitted, and in order to perform statistical analysis 142 telomeres per cell cycle phase were

randomly selected for analysis. In Figure 4 the average MSD curves are shown for the different phases of the cell cycle. First a Mixed Model analysis was performed using SPSS in order to determine whether there was a group effect of the telomeres that were measured within a cell, for both the MSDt and the MSDp values. The group-effect was not significant in both the MSDt and the MSDp measurements, so subsequently a One-Way Anova analysis was performed on both. For the MSDp (the distance travelled by a telomere) we found that telomeres travel over a significantly larger distance in the G1-, the G2- and the beginning of the S-phase (BegS) than during the middle (MidS) or late part (LateS) of the S-phase. For the MSDt, the nuclear area in which a telomere moves, telomeres were found to move in a significantly greater volume during the G1- and the G2 phase than during the entire S-phase.

Discussion

This paper describes the basics and features of the program STACKS, which was developed for the tracking of nuclear particles. It has been shown that this program provides sufficient visual feedback of the processing steps involved. It offers the user adequate options and flexibility to analyze data sets of varying quality. There are also tools to interactively correct for improper automatic segmentation and even when particles are incorrectly tracked, the tracks can still be corrected by the user so that finally the proper measurements can be obtained. Additional features and measurements have been built in to make the program suitable for other applications as well.

An approach was chosen, where only the visible images are kept in memory, so that no special requirements are necessary to run the program on a regular PC. It is shown that this has hardly any impact on the responsiveness of displayed stacks. However, it gives some overhead (in the order of a few seconds) when a complete stack is processed, which increases to about a minute for a stack of 1000 images of 1024 x 1024 pixels. This overhead-time will be reduced with almost 50% when a SSD disk is applied, and probably similar results would be obtained when disks would be put in RAID. It should be realized that neither the PC used in this paper, nor the GPU board and the SSD drive are nowadays the fastest on the market. When a new system would have to be assembled as of today, performance would even be better.

The program provides a number of image operations based on the GPU. They were programmed using two different shader languages, namely HLSL (using DirectX) and CUDA. For most functions the HLSL implementation is somewhat more efficient, especially for the operations on color images. It appears also that the time to transfer images between the GPU board and computer memory is also faster using DirectX and HLSL. On the other hand the fft library from CUDA is more efficient than the fft functions written in HLSL and the convolution filter in CUDA, which is based on local memory, is also faster than the implementation using global texture memory in HLSL. The operations programmed in software carried out by the CPU are always slower compared to GPU processing despite the overhead of transferring images to and from the GPU board. The use of GPU programming especially for the processing of 4D stacks is extremely useful as more than thousand images have to be processed thereby reducing the total processing time from minutes to seconds.

The market for GPU boards is dominated by two contenders, namely Ati-Amd and Nvidia. Ati-Amd has also made a software development kit available for general purpose GPU processing, called Stream. Unfortunately, both CUDA and Stream are dedicated to the hardware of the specific vendor, so that the shader programs cannot be exchanged. A future choice could be to make use of OpenCL from Apple and DirectCompute from Microsoft, which will offer support for GPU boards of both vendors. Currently there is no support in STACKS for hardware acceleration using Ati boards. As all image operations in STACKS are also written in CPU based software, it is still possible to run STACKS anyway. However, it was shown that GPU based processing provides a significant speed improvement for image operations especially when the kernel size becomes larger.

STACKS can operate directly on the image files derived from commercially available Zeiss and Leica microscope systems. It is, however, also possible to read images from a folder. By renaming independent 2D images in a way that the program “thinks” that they form a 3D or

4D stack, it is possible to analyze large image sets of independent images using STACKS. In this way large image sets have already been analyzed using STACKS.

Future Developments

The current version of STACKS supports the tracking of maximal 255 objects as object labeling was originally developed for 8 bit images only. This is sufficient for tracking telomeres in one cell at a time, but when for instance viral particles have to be tracked in larger images, this will become a bottleneck. Object labeling has already been extended to 16 bit and the tracking in 16 bit images and thus tracking more than 65000 objects will be realized in the near future.

The number of stacks that can be simultaneously handled by the program is currently fixed to 4 grey-value 4D stacks, 4 color 4D stacks, a binary and a label 4D stack with the same amount of 3D stacks, when a maximum projection is performed. This limitation is mainly caused by the fact that only those stacks are foreseen in the menus and dialogues. In a future release those limitations will be removed by making the menus dynamic and by providing facilities to create additional stacks. In this way a larger number of stacks can be opened simultaneously, as long as memory and disk space allow it.

Currently all image operations based on the GPU are 2D operations. 3D Object labeling and 3D watershed is carried out by normal CPU processing and the GPU based nearest neighbour deconvolution takes only the slice below and above into account. However, the latest release of CUDA supports 3D textures which will ease the development of 3D image operations, such as 3D Min / Max operations, 3D convolutions and 3D distance transforms. It can also help with more complex operations such as 3D deconvolution. A new library with the 3D fast Fourier is also released by Nvidia, so that 3D deconvolution based on inverse filtering should be relatively easy to implement.

Up to now only one graphic board is supported by STACKS. It is possible to have more GPU boards in a PC or additional special boards for general GPU processing like the Tesla boards from Nvidia. This can boost the performance of GPU processing even further. STACKS would be ideally suited for this approach, as the processing of different 2D or 3D images could be easily be carried out in parallel. This possibility will be explored in future. Finally it should be mentioned that the program STACKS is free of charge available on request for non-commercial use.

Acknowledgements

We like to thank Ron Wolterbeek from the Department of Medical Statistics and Bioinformatics of the LUMC for assisting with the statistical analysis of the MSD measurements. The work for development of image processing functions using the GPU was financially supported by Genetix, New Milton, UK.

References

Adrian R.J. 2005. Twenty years of particle image velocimetry. *Exp. Fluids* **39,** 157-483

Ashkin A.1997. Optical trapping and manipulation of neutral particles using lasers. *Proc. Natl. Acad. Sci. U S A* **94,** 4853-4860

Bacher C.P., Reichenzeller M., Athale C., Herrmann H. & Eils R. 2004. 4-D single particle tracking of synthetic and proteinaceous microspheres reveals preferential movement of nuclear particles along chromatin – poor tracks. *BMC Cell Biol.* **5**, 45

Bausch A.R., Ziemann F., Boulbitch A.A., Jacobson K. & Sackmann E. 1998. Local measurementsof viscoelastic parameters of adherent cell surfaces by magnetic bead microrheometry. *Biophys. J.* **75,** 2038-2049

Carter B.C., Shubeita G.T. & Gross S.P. 2005. Tracking single particles: a user-friendly quantitative evaluation. *Phys. Biol.* **2,** 60-72

Cheezum M.K., Walker W.F. & Guilford W.H. 2001. Quantitative comparison of algorithms for racking single uorescent particles. *Biophys. J.* **81,** 2378 – 2388

Grant I. 1997. Particle image velocimetry: a review. *Proceedings of the Institution of Mechanical Engineers, Part C: J. Mech. Eng. Sci.* **211,** 55-76

Crocker J.C. & Grier D.G. 1996. Methods of digital video microscopy for colloidal studies. *J. Coll. Interf. Sci.* **179,** 298-310

Csink A.K. & Henikoff S. 1996. Genetic modification of heterochromatic association and nuclear organization in Drosophila. *Nature* **381**, 529-531

Dernburg A.F., Broman K.W., Fung J.C., Marshall W.F., Philips J., Agard D.A. & Sedat J.W. 1996. Perturbation of nuclear architecture by long-distance chromosome interactions. *Cell* **85**, 745-759

Farrugia J-P., Horain P., Guehenneux E. & Allusse Y. 2006. GPUCV: A framework for image processing acceleration with graphics processors. *Proc. of the IEEE International Conference on Multimedia & Expo (ICME 2006), July 9-12, Toronto, Ontario, Canada.*

Francastel C., Schubeler D., Martin D.I. & Groudine M. 2000. Nuclear compartmentalization and gene activity. *Nat. Rev. Mol. Cell Biol.* **1**, 137-143

Leonhardt H., Rahn H.-P., Weinzier P., Sporbert A., Cremer T., Zink D. & Cardoso M.C. 2000. Dynamics of DNA Replication Factories in Living Cells. *J. Cell Biol.* **149,** 271-280

Manuelidis L. 1985. Individual interphase chromosome domains revealed by in situ hybridization. *Hum. Genet.* **71**, 288-293

Miura K. 2005. Tracking movement in cell biology. *Adv. Biochem. Eng. Biotechnol.* **95,** 267-295

Moreland K. , Angel E. 2003. The FFT on a GPU. *SIGGRAPH Eurographics Workshop on Graphics Hardware.* 112–119

Platero, J., Csink, A., Quintanilla, A. & Henikoff, S. 1998. Changes in chromosomal localization of heterochromatin binding proteins during the cell cycle in Drosophila. *J. Cell Biol.* **140**, 1297-1306

Podlozhnyuk V. Image Convolution with CUDA. 2001. CUDA SDK code samples. www.nvidia.com

Rong G. & Tan. T-S. 2006. Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D 2006), Redwood city, CA, USA, March 2006*. 109-116

Ruifrok A.C. & Johnston D.A. 2001. Quantification of histochemical staining by color deconvolution. *Anal. Quant. Cytol. Histol.* **23**, 291-299

Sbalzarini I.F. & Koumoutsakos P. 2005. Feature point tracking and trajectory analysis for video imaging in cell biology. *J. Struct. Biol.* **151,** 182-195

Spector D.L. 2003. The dynamics of chromosome organization and gene regulation. *Annu. Rev. Biochem.* **72**, 573-608

Strzodka R. & Ihrke I., Magnor M. 2003. A graphics hardware implementation of the Generalized Hough Transform for fast object recognition, scale, and 3d pose detection. *Proc. of IEEE Int.Conf. on Image Analysis and Processing (ICIAP'03)*, 188–193

Strzodka R., Telea A. 2004. Generalized Distance Transforms and skeletons in graphics hardware. *Proc. of EG/IEEE TCVG Symposium on Visualization (VisSym '04), 221–230.*
Fung J., Mann S., Aimone C. 2005. Openvidia: Parallel gpu computer vision. Proc. of the ACM Multimedia 2005, November 2005, 849–852

Trask B., van den Engh G., Pinkel D., Mullikin J., Waldman F., van Dekken H. & Gray J. 1988. Fluorescence in situ hybridization to interphase cell nuclei in suspension allows flow cytometric analysis of chromosome content and microscopic analysis of nuclear organization. *Hum. Genet.* **78**, 251-259

Tseng Y., Kole T.P. & Wirtz D. 2002. Micromechanical mapping of live cells by multiple-particle-tracking microrheology. *Biophys. J.* **83,** 3162-3176

Verschure P.J., van Der Kraan I., Manders E.M. & van Driel R. 1999. Spatial relationship between transcription sites and chromosome territories. *J. Cell Biol.* **147**, 13-24

Waigh T.A. 2005. Microrheology of complex fluids. *Rep. Progr. Phys.* **68,** 685-742

Walter J., Schermelleh L., Cremer M., Tashiro S. & Cremer T. 2003. Chromosome order in HeLa cells changes during mitosis and early G1, but is stably maintained during subsequent interphase stages, *J. Cell Biol.* **160**, 685–697

Weihs D., Mason T.G. & Teitell M.A. 2006. Bio-microrheology: a frontier in microrheology. *Biophys. J.* **91,** 4296-4305

Figures

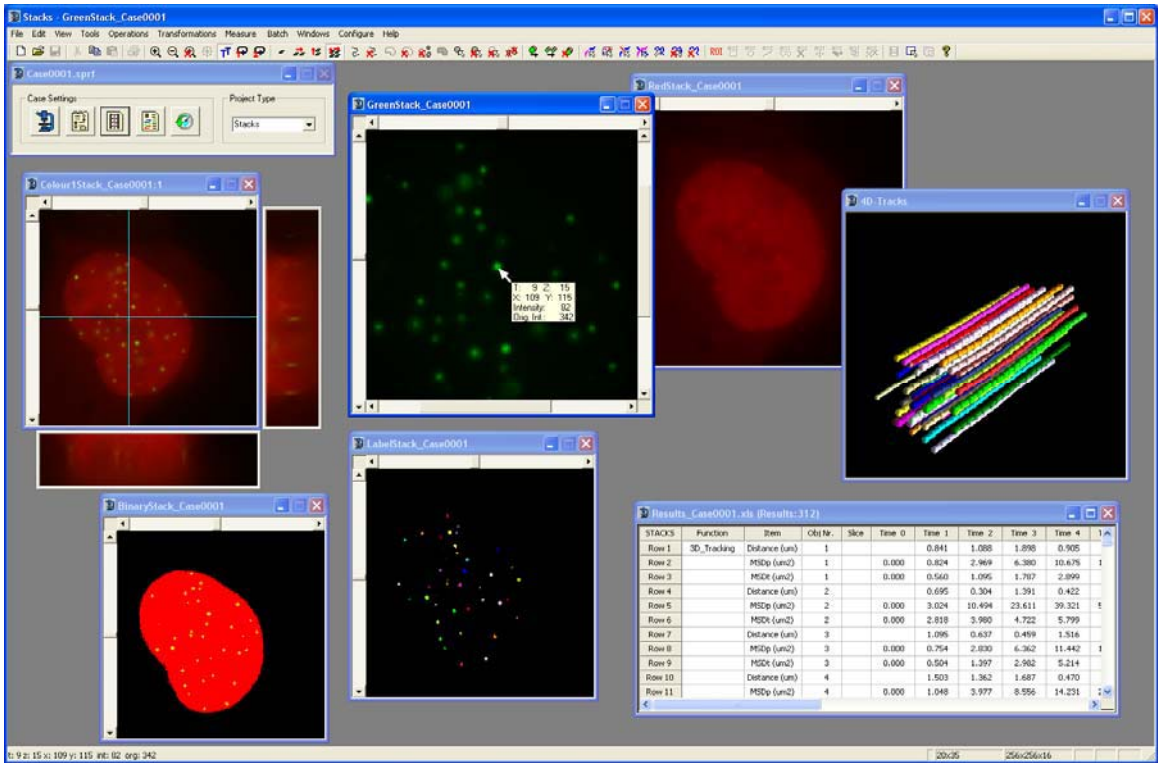


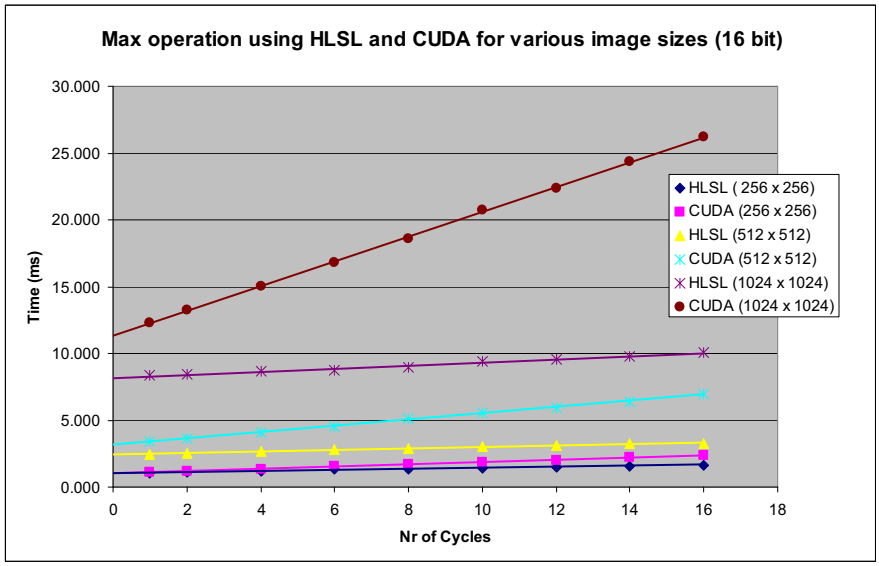
Figure 1. The desktop of STACKS is shown with the windows of the various stacks. Note that the window of the color stack is expanded with the cross-sections at the position of the crosshair pointer. The Green stack is displayed enlarged. This adds scroll bars to the bottom and the right for positioning the image. The scrollbar at the top is reserved for the time points and at the left for the z-slices. It is possible to show intensity information at the position of the cursor as is shown in the window of the Green stack.

Dimensions Stack	Samsung disk			OCZ SSD		
	Create stack (s)	Read stack (s)	Read / Write stack (s)	Create stack (s)	Read stack (s)	Read / Write stack (s)
25x40x256x256	5.4	0.20	3.75	4.7	0.18	2.50
25x40x512x512	25.9	1.01	14.5	16.9	0.98	8.83
25x40x1024x1024	137.1	4.19	54.80	88.7	4.10	27.83

Table 1. This table shows the time needed to create a new stack when it is opened for the first time, to read an existing stack only and to read an existing stack and write it back after processing. Reading the stack only occurs when the user scrolls through the stack. The last column is an indication for the disk overhead that is involved during processing. Using a solid state disk a significant improvement is obtained.

Operation	Bit Depth	HLSL (ms)	CUDA (ms)	Software CPU (ms)
Erosion (cycles: 8)	8	2.4	5.1	65.8
	16	2.8	4.9	66.1
	24	4.8	10.2	184.1
Opening (cycles: 8)	8	2.8	6.6	113.5
	16	2.9	6.9	113.8
	24	5.7	10.2	324.4
Sobel filter	8	2.0	2.7	11.4
	16	2.5	3.1	13.5
	24	4.2	8.9	30.2
LaPlace filter	8	2.0	2.7	5.3
	16	2.5	3.1	6.4
	24	4.1	8.8	13.6
Convolution (kernel:21x21)	8	6.2	4.1	118.1
	16	6.7	4.4	115.2
	24	17.2	11.5	387.9
Median (kernel: 3x3)	8	2.1	3.7	151.2
	16	2.6	4.0	153.4
	24	4.3	12.1	354.1
Kuwahara (kernel: 5x5)	8	5.8	5.4	182.7
	16	5.8	5.6	181.3
	24	7.2	16.7	533.2
Unsharp Mask (kernel: 7x7)	8	2.2	3.3	80.5
	16	2.9	3.7	78.4
	24	4.7	11.7	249.5
FFT spectrum	8	7.9	4.4	230.2
	16	8.6	4.8	228.4
Butterworth band pass (FFT)	8	11.9	8.1	not implemented
	16	13.9	8.3	not implemented
Gaussian low pass (FFT)	8	11.8	8.5	not implemented
	16	13.7	8.8	not implemented
RGB->HSI	24	4.0	8.2	43.4
Component Blend	24	4.3	8.7	13.3
Colour Deconvolution	24	4.7	8.8	127.9
Distance Transform	8	6.5	17.6	27.1
Nearest Neighbour (32 slices)	8	189.4	214.9	2772.6
	16	522.3	535.2	2798.3
	24	308.6	495.8	8597.7

Table 2. This table gives an overview of the time needed to process one image of 512 x 512 pixels for various image operations using the GPU and software. The overhead of transferring images to and from the GPU is included for all operations. For the nearest neighbour deconvolution the time to process 32 slices is given and the overhead to transfer the images to and from disk is included for this operation as well.



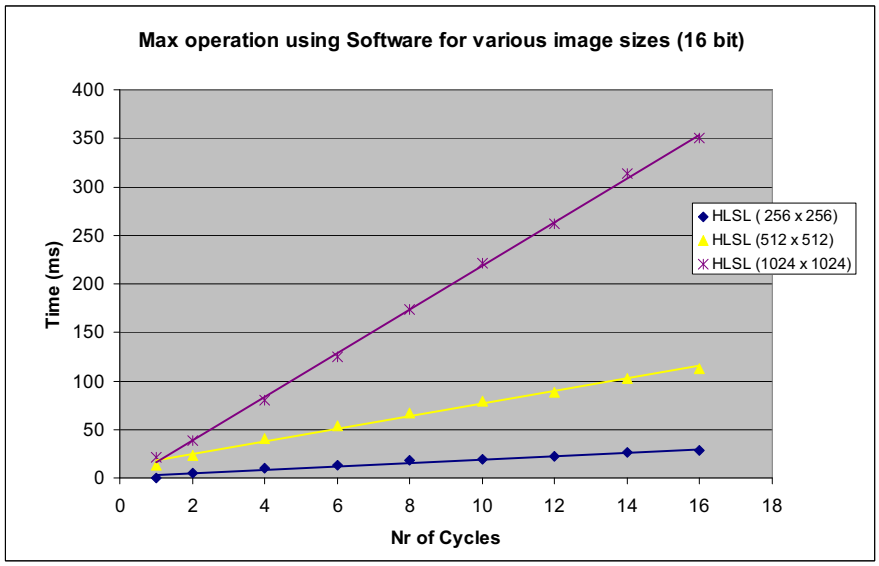


Figure 2. Figure 2A shows the MAX operation as function of the number of cycles for the HLSL shader and the CUDA shader program. HLSL is always faster. The time to transfer an image to/from the GPU board is included. By extrapolating the function to the y-axis the time necessary for image transfer is obtained. In figure 2B the equivalent is shown for the software implementation using the CPU.

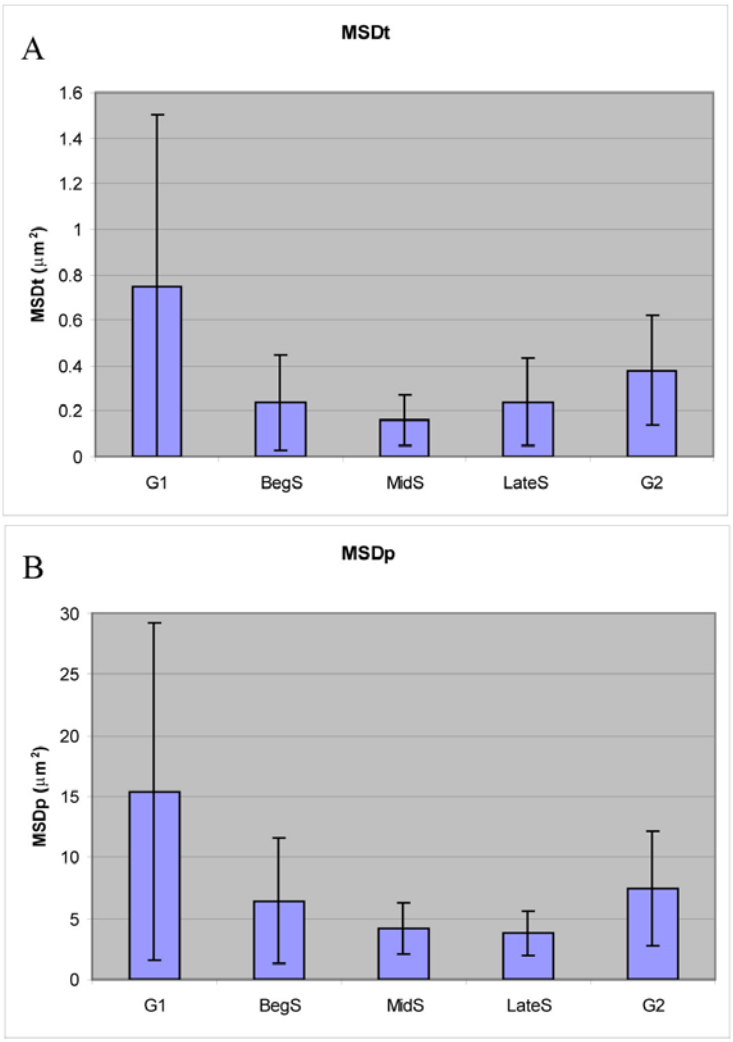


Figure 3. MSDt and MSDp graphs. A) The average values of the mean squared displacement (MSDt) and the standard deviation for the area in which the telomeres move in U2OS cells are shown. B) MSDp: Distance travelled by telomeres during different phases of the cell cycle.

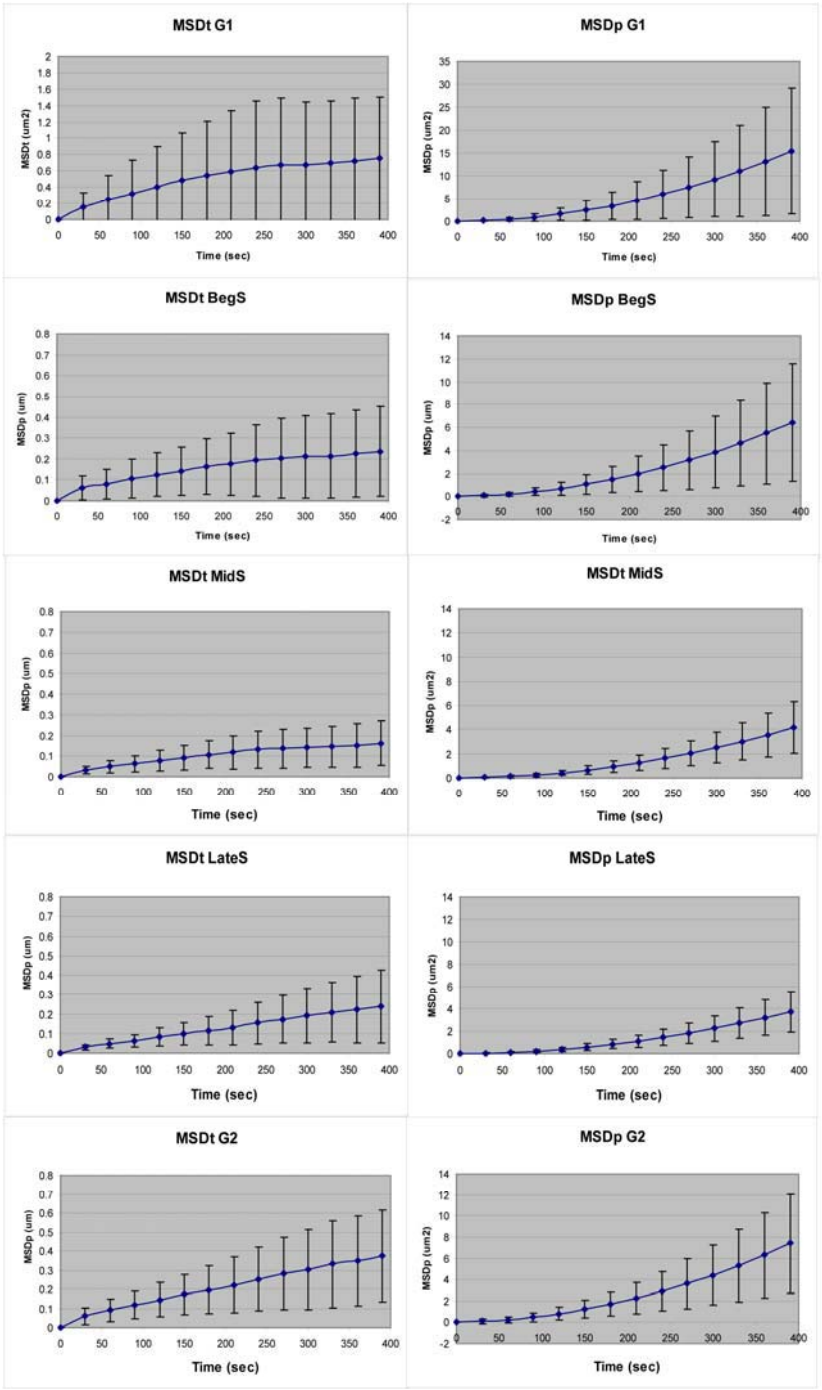


Figure 4. The MSDt and the MSDp curves are shown for the different phases of the cell cycle. A total of 6 cells and 142 telomeres were tracked per cell using the STACKS tracking software. Error bars represent the variance. Note that, as the telomeres become more dynamic, the error bars are also increasing. This may indicate that particularly during the G1 phase the MSD values show strong variance. This variance during the G1 phase of the cell cycle may very well be caused by a change of telomere dynamics during the G1 phase. A likely explanation could be that telomeres are very dynamic during the beginning of the G1 phase and that their dynamics decrease as they approach the end of the G1 phase. This hypothesis is in compliance with the idea that chromatin is very dynamic immediately after mitosis, because at that time many chromatin rearrangements take place (Walter et al., 2003). It is possible that for the same reason telomeres are increasingly dynamic in the G2 phase, indicating that just before mitosis also chromatin rearrangements may occur at a higher frequency.