# MotionCor2 User Manual

**Shawn Zheng**
**University of California San Francisco**

**Version: 1.4.4**
**Release Date: 08/11/2021**

## General

MotionCor2 is a multi-GPU program that corrects beam-induced sample motion recorded on dose fractionated movie stacks. It implements a robust and efficient iterative alignment algorithm that delivers precise measurement and correction of both global and local motions at single pixel level, suitable for both single-particle and tomographic images. MotionCor2 is sufficiently fast to keep up with automated data collection. The result is an exceptionally robust strategy that can work on a wide range of data sets, including those very close to focus or with very short integration times. Application significantly improves Thon ring quality and 3D reconstruction resolution. MotionCor2 is a comprehensive program that integrates gain correction, detection and correction of individual and cluster of bad pixels, dose weighting, and supports both MRC, TIFF, and EER file formats.

MotionCor2 is free for academic use and can be downloaded from:
**http://msg.ucsf.edu/software**

**Contacts:**
Suggestions, discussions, and technical support:
**Shawn Zheng: szheng@msg.ucsf.edu**

**Licensing MotionCor2:**
David Agard: agard@msg.ucsf.edu
Yifan Chen: YCheng@ucsf.edu

**Table of Content**

## 1. Installation and system requirement

MotionCor2 is a GPU accelerated program that runs on Linux platform equipped with one or more advanced nVidia GPU cards. The current version was compiled on Centos 7. Various executables are provided for CUDA 10.0, 10.1, 10.2, 11.1, 11.2, and 11.3, respectively. CUFFT and LibTIFF are required.

MotionCor2 is a single-program package. Once unpacked, it is ready to go should correct libraries be installed properly. MotionCor2 supports MRC, TIFF, and EER files where movies are saved.

### 1.1 CPU memory

MotionCor2 buffers as many frames as possible in GPU memory, which is, very often, far from enough to hold the entire gain-corrected movie. The remaining frames have to stay in CPU memory. For this sole reason, the capacity of CPU RAM dictates the sizes of movies that can be processed. It is important to estimate how much CPU memory is needed given the typical sizes of movies routinely collected or the upper limit of movie sizes given the CPU memory of an existing system. For simplicity and certainty, our estimate assumes all movie frames on CPU side. Gain corrected movie needs 4Byte/pixel. As a result, a 100-frame K3 super-res movie occupies 36GB memory after gain correction. A single MotionCor2 process roughly allocate 1.25 x 36 = 45GB memory if it runs in single mode. When running in batch mode, see section 12, MotionCor2 loads a second movie (1B/pixel) as soon as the first one gets processed. In most cases there are two movies in CPU memory at the same time, one gain corrected and one not. Therefore, the system is recommended to have 45 + 9 = 54GB memory. We would also like to have an extra 100GB memory for other processes.

## 2. Quick start

MotionCor2 is a command-line program configurable by means of command-line parameters. The following is the minimum configuration to run the program.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc
```

With this configuration MotionCor2 corrects only the global motion. It loads the movie "Stack_0001.mrc" and searches for the gain reference in the extended header of Stack_0001.mrc. If found, the gain reference is loaded and applied to each frame. If not, the program proceeds without gain correction. The global motion is then measured and corrected by phase shift in Fourier space.

The minimum configuration for MotionCor2 to correct both global and local motion is as follows.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Patch 5 5
```

MotionCor2 first corrects the global motion for each frame and then divides the corrected frames into 5x5 patches on which the local motion is measured. Once completed, the distance-based scheme is used to interpolate local motions at individual pixels, which are then corrected in real space. For K3 movie, it is recommended to use -Patch 7 5 instead.

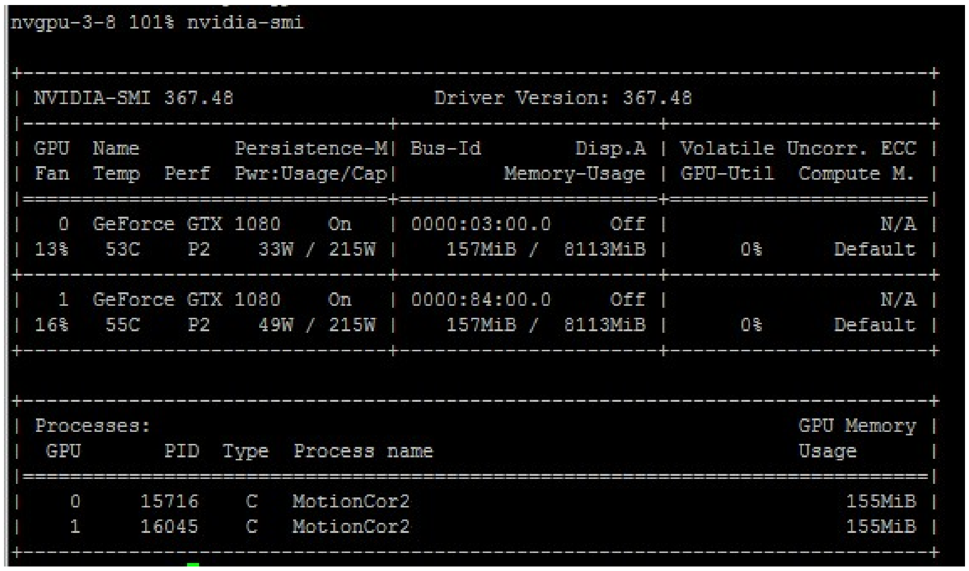
## 3. Running on multiple GPUs

MotionCor2 can run on multiple GPUs by distributing computation onto each participating GPU. A faster GPU will be assigned more computation than the slower ones. The following example shows the configuration of using 4 GPUs.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Patch 5 5 \
-Gpu 0 1 2 3
```

In this example the hosting node has 4 GPUs installed with unique IDs of 0 1 2 3. The GPU IDs can be found by running a nVidia program nvidia-smi on the command line. GPU 0 will be used if -Gpu does not appear on the command line.

It is very important to note that each GPU can only be used by one MotionCor2 process. If any GPU is shared across multiple processes, motion correction may fail and the system many hang! It is strongly recommended to use nvidia-smi to check before starting MotionCor2. The following is an example of results reported by nvidia-smi.



```
nvgpu-3-8 101% nvidia-smi

+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.48                 Driver Version: 367.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1080    On   | 0000:03:00.0     Off |                  N/A |
| 13%   53C    P2    33W / 215W |    157MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1080    On   | 0000:84:00.0     Off |                  N/A |
| 16%   55C    P2    49W / 215W |    157MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     15716    C   MotionCor2                                     155MiB |
|    1     16045    C   MotionCor2                                     155MiB |
+-----------------------------------------------------------------------------+
```

Fig. 1 nvidia-smi shows that two GPUs, 0 and 1, are installed. Both of them are currently used by a MotionCor2 process.

### 3.1. Use a subset of GPUs

In case multiple MotionCor2 processes are needed to run side by side, -Gpu and -UseGpus can be used together to split all free GPUs over all processes. -Gpu should be followed by the IDs of installed GPUs and -UseGpus specifies how many GPUs will be used by a process. MotionCor2 maintains a text file MotionCor2_FreeGpus.txt in /tmp to track all free GPUs. For a system has four GPUs installed, for example, we can start two processes using -Gpu 0 1 2 3 -UseGpus 2. Each process chooses 2 free GPUs to run. When all GPUs are in use, the next process then cannot be started.

## 4. Apply gain reference

Gain reference can be saved either in the extended header of MRC files or as a separate MRC file. The following example shows how to specify the gain reference on the command line.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Gain /home/data/MyGainRef.mrc \
-Patch 5 5
```

In this example MotionCor2 loads the gain reference from "MyGainRef.mrc". If this file is not found, the extended header of "Stack_0001.mrc" will be checked for gain reference. If gain reference is not found, MotionCor2 proceeds without gain correction.

### 4.1. Rotate and flip gain reference

Gain reference is usually collected in accordance with the chip orientation. Movies can be rotated and/or flipped. MotionCor2 give users options to rotate/flip the gain reference to match the orientation of collected movies by means of -RotGain and -FlipGain.

-RotGain: rotate gain reference counter clockwise. It takes four values from 0 to 3.

0 – no rotation, default,
1 – rotate 90°,
2 – rotate 180°,
3 – rotate 270°.

-FlipGain: flip the gain reference.

0 – no flipping, default,
1 – flip upside down (flip around horizontal axis),
2 – flip left right (flip around vertical axis).

If both -RotGain and -FlipGain are enabled, the gain reference will be rotated first and flipped next.

## 5. Subtract dark reference

Dark reference can be loaded from a MRC file. If it is loaded successfully, dark reference will be subtracted from each frame before gain reference is applied. The following example shows how to specify dark reference on the command line.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Gain /home/data/MyGainRef.mrc \
-Dark /home/data/MyDarkRef.mrc \
-Patch 5 5
```

We assume dark and gain references bear the same orientation. As a result, -RotGain and -FlipGain settings are also applied to the dark reference.

## 6. Alignment configuration

Users can set the number of iterations and the tolerance for alignment error. The corresponding parameters are highlighted in red in the following example.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Gain /home/data/MyGainRef.mrc \
-Dark /home/data/MyDarkRef.mrc \
-Patch 5 5 \
-Iter 10
-Tol 0.5
```

In this example the iterative alignment procedure terminates when either the alignment error is less than 0.5 pixel or 10 iterations have reached.

## 7. Discard initial and final frames

The following example shows how to throw away 2 starting frames and 3 frames at the end. The discarded frames are neither included in alignment nor in the corrected sum.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Gain /home/data/MyGainRef.mrc \
-Dark /home/data/MyDarkRef.mrc \

-Patch 5 5 \
-Iter 10 \
-Tol 0.5 \
-Throw 2
-Trunc 3
```

If not specified, all frames are included.

## 8. Align to specific frame

By default, a movie stack is aligned to its central frame. However, there are some occasions when users want to align their stacks to a specific frame. This is can be done by specifying the frame number after –FmRef. The frame number is zero-indexed based upon the loaded frames.

## 9. Dose weighting

MotionCor2 implemented the dose weighting scheme developed by Grant et al. [1]. The following example shows how to enable dose weighting.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Gain /home/data/MyGainRef.mrc \
-Patch 5 5 \
-Iter 10 \
-Tol 0.5 \
-Throw 2 \
-Kv 300 \
-PixSize 0.5 \
-FmDose 1.2 \
```

Users need to specify the high tension in kV, and the pixel size of the input movie in angstrom, and the frame dose in e/Å$^2$. If any of the three parameters is missing, dose weighting is skipped. When dose weighting is enabled, both dose-weighted and unweighted sums are generated. The weighted sum is saved in the file with its name appended with "_DW" as CorrectedSum_DW.mrc in the aforementioned example.

### 9.1 Generate dose weighted sum of selected frames

MotionCor2 automatically generates a third sum of selected frames that fall within the user-specified dose range. The default range "–SumRange 3.0 25.0" sums the frames whose accumulated doses fall within the range from 3 to 35 e/A$^2$. The MRC file storing this sum has the file name appended with "_DWS".

This option can be turned off if the zeroes are specified.

### 9.2 Movies acquired with variable frame exposure

Data collection with variable frame exposure (VFE) is used to reduce blurring in early frames resulting from much stronger early motion. In the past when frame exposure was fixed, 3-5 early frames are typically excluded. However, the abandoned early frames are the least radiation damaged and would bear high-resolution information if not blurred by the stronger beam-induced motion. VFE uses shorter exposures to mitigate the blur in early frames and longer exposures for later ones to reduce the movie size. It has been made available recently in SerialEM. However, each frame in a VFE movie no longer receives the same dose. As a result, the single input of frame dose following –FmDose would be unable to correctly determine the accumulated dose. In a joint effort to support Thermal Fisher EER movies, MotionCor2 added a new option –FmIntFile that stands for Frame Integration File, a 3-column text file for this purpose. The first column lists the number of frames that have the same exposure. The second column lists number of raw frames being summed into rendered frames. The third column is the dose a raw frame receives during its exposure. A raw frame refers to the frame in a movie file to be processed by MotionCor2. A rendered frame is the simple sum of one of more raw frames. From now on, motion correction is performed on rendered frames instead of raw frames. If –FmIntFile does not appear on the command line, rendered frames are the same as raw frames.

Here is an example assuming a movie is acquired using three frame exposures. Frame 1 to 20 is acquired at 20 ms/frame, 21 to 60 at 40 ms/frame, and 61 to 140 at 60 ms/frame. Note that the dose received by each frame can be calculated given dose rate, pixel size, and frame exposure. For simplicity, let's assume we have already calculated the dose for frame 1, 0.15 e/A$^2$. The frame integration file following –FmIntFile should be as follows:

```
20   1   0.15
40   1   0.30
80   1   0.45
```

The second column indicates the input movie is rendered as it is and the third column lists the dose distribution over the frames. In this case, –FmDose will be ignored. Users still need to provide pixel size (–PixSize) and high tension (–Kv) in order to enable dose weighting.

## 10. Correct anisotropic magnification

Anisotropic magnification causes images less magnified in one direction (major axis) and more magnified in the direction (minor axis) perpendicular to major axis. MotionCor2 corrects anisotropic magnification by stretching the image along the major axis. Users need to obtain the parameters of anisotropic magnification using Tim Grant's program **mag_distortion_estimate** [2]. These parameters can then be provided to MotionCor2 using "-Mag" that is followed by major scale, minor scale, and the angle of the major scale. In the following example MotionCor2 corrects the anisotropic magnification that has major scale of 1.003, minor scale of 0.998, and 34.0° of distortion angle reported by mag_distortion_estimate.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Gain /home/data/MyGainRef.mrc \
-Patch 5 5 \
-Iter 10 \
-Tol 0.5 \
-Throw 2 \
-Kv 300 \
-PixSize 0.5 \
-FmDose 1.2 \
-Mag 1.003 0.998 34.0
```

**Note:** In the previous versions of MotionCor2, "-Mag" should be followed by magnifications rather than scales. Therefore, users need to invert the scales to magnifications.

## 11. Image binning - Fourier cropping

Image binning is implemented by cropping in Fourier domain. -FtBin can be used to bin the motion-corrected image to a specified resolution. Values for this option can be either an integer or a float that is bigger than 1. In the following case, the output image is cropped in Fourier space by 1.5x.

```
MotionCor2 -InMrc /home/data/Stack_0001.mrc \
-OutMrc /home/data/CorrectedSum.mrc \
-Gain /home/data/MyGainRef.mrc \
-Patch 5 5 \
-Iter 10 \
-Tol 0.5 \
-Throw 2 \
-FtBin 1.5
```

If the raw movie stacks are collected in super-resolution mode and the final images is intended to be binned, we recommend to use the super-resolution stacks as input and let MotionCor2 do the binning. This is a better practice than passing binned stack to MotionCor2. Since the local motion is corrected by linear interpolation that has low-pass effect in Fourier space, it is preferred to correct the local motion at super-resolution pixel to minimize the loss of high-frequency information due to interpolation.

## 12. Batch processing

Batch processing overlaps the disk operation with the intensive computation involved in motion correction. As a result, the disk I/O time is almost completely shadowed by the computational time. MotionCor2 automates the sequential motion correction of multiple single-particle movie stacks based upon pattern recognition of file names. "-Serial 1" enables the batch processing. There are two scenarios. First, when the folder contains only the movie stacks, the following examples shows the how to configure the command line.

```
MotionCor2 -InMrc /home/data/ \
-OutMrc /home/Sum/Corrected \
-Gain /home/Ref/MyGainRef.mrc \
-Patch 5 5 \
-Iter 10 \
-Tol 0.5 \
-Throw 2 \
-Kv 300 \
-PixSize 0.5 \
-FmDose 1.2 \
-FtBin 2 \
-Gpu 0 1 \
-Serial 1
```

In this case all the MRC files in "/home/data/" are treated as movie stacks and corrected sequentially. The corrected sums are named by prefixing "Corrected" to the input file names and saved in "/home/Sum/" directory.

Second, if the input folder contains mixed files, the following example shows how to configure the command line for the program to choose only the MRC stack files.

```
MotionCor2 -InMrc /home/data/Stack_ \
-InSuffix Raw.mrc \
-OutMrc /home/Sum/Corrected \
-Gain /home/Ref/MyGainRef.mrc \
-Patch 5 5 \
-Iter 10 \
-Tol 0.5 \
-Throw 2 \
-Kv 300 \
-PixSize 0.5 \
-FmDose 1.2 \
-FtBin 2 \
-Gpu 0 1 \
-Serial 1
```

MotionCor2 chooses only the files with names prefixed with "Stack_" and suffixed with "Raw.mrc" in "/home/data" directory. Here are two examples, "Stack_1234Raw.mrc" and "Stack_3456-Raw.mrc".

## 13. Support for TIFF files

-InTiff is used specify an input of TIFF file. Currently, there is no support for batch processing of TIFF files. The support of TIFF files is limited to single-particle movie stacks.

```
MotionCor2 -InTiff /home/data/Stack_0001.tif \
-OutMrc /home/Sum/Corrected_0001.mrc \
-Gain /home/Ref/MyGainRef.mrc \
-Patch 5 5 \
-Iter 10 \
-Tol 0.5 \
-Throw 2 \
-Kv 300 \
-PixlSize 0.5 \
-FmDose 1.2 \
-Gpu 0 1
```

## 14. Support for EER files

Version 1.4.0 starts to support EER movies. –InEer has been added for users to input EER movies. In addition, users also need to specify EER sampling using –EerSampling followed by a value from 1, 2, and 3 corresponding to 1x, 2x, and 4x upsampling, respectively. Since an EER movie usually contains hundreds of raw frames, motion correction on the original movie would become very inefficient, if not impossible due to the limitation of system memory. Therefore, we recommend using –FmIntFile to convert the input movie into the rendered one that contains, ideally, less than 200 rendered frames. As described in section 9.2, –FmIntFile expects a three-column text file. The following is an exemplary file for converting an EER movie of 567 EER frames into the rendered one containing 103 rendered frames. Again, a rendered frame is a sum of multiple raw frames, i.e. the EER frames in this case.

```
120   3   0.1
447   7   0.1
```

The first line converts 120 EER frames into 40 rendered frames and the second line yields 63 rendered frames. Users can add more lines if the conversion of finer granularity is needed. The last column is the dose in e/A$^2$ each EER frame receives. Motion correction is done on the rendered frames.

## 15. Output motion corrected stack

Motion corrected stacks can be generated by specifying "-OutStack 1" along with the motion corrected sum. Note that in this setup the dose weighting step is skipped. As a result, the corrected sum and stack are not dose weighted. The output stack is stored in a MRC file with "_Stk" appended to the end of the output file name. This option is not available for dose fractionated tomographic tilt series.

As of version 1.4.4, this option has a second parameter that specifies binning in z, i.e, number of frames to be summed per output frame in the corrected stack. "-OutStack 1 4", for example, means the output frame is a sum of 4 motion-corrected frames.

## 16. Low-signal movies

There are two parameters users can play with. The first one is B-factor. Its value can be changed by -Bft. Since version 1.1.0, -Bft takes two parameters of which the first one is used in global-motion measurement and the second is for local-motion.

### 16.1 Group frames to enhance signal

Another parameter is to adjust the setting of -Group whose default value is 1. For movies with low signal to noise ratio, a bigger value has been found quite effective. -Group instructs the program to equally divide the input stack into non-overlapping sub-groups. Instead of aligning individual frames, the sums of these sub-groups are aligned. The shifts of individual frames are then interpolated and extrapolated. For example, -Group 3 divides the input stack of 120 frames into 40 sub-groups, each containing 3 frames. As opposed to increasing B-factor, this is a recommended approach.

### 16.2 Non-uniform grouping

When a raw movie is rendered by means of -FmIntFile, non-uniform grouping will be automatically invoked if -Group appears on the command line. There are two cases that involve non-uniform grouping. The first one is the frame integration file has the second column filled with different values. Assume an integration file has the following content and -Group 4 appears on the command line.

```
20   1   0.3
40   2   0.3
80   4   0.3
```

The first 20 frames will be grouped every 4 frames. Frame 21 to 60 will be grouped every 2 frames. The last 80 frames will not be grouped. Motion measurement is then performed on the (5 + 20 + 80 = 105) group summed images. EER movies or movies collected at high frame rate fall into this case.

Movies collected using variable frame exposure fall into the second case where the second column of the frame integration file filled with 1s but the dose in the third column varies. It is the third column that will be used to determine the non-uniform grouping. Again, assume a frame integration file as follows and -Group is set to 4.

```
20   1   0.3
40   1   0.6
80   1   1.2
```

In this case, the first 20 frames will be grouped every 4 frames. Since each of the next 40 frames receives two times of the dose as opposed to the first 20 frames, they will be grouped every 2 frames. The last 80 frames will not be grouped since their frame dose is 4 times of the first 20 frames.

## 17. Correct camera defects

In addition to dynamically detect and correct defects in acquired movie stacks, users can specify fixed regions of defects in a text file. This file is composed of multiple lines of which each contains four space-separated integers, x, y, w, and h that define a rectangular region of defects. x and y are the pixel coordinates of the lower left corner of such a region where w and h denote the width and height, respectively. The full path of this text file should follow the tag "-DefectFile". The defective pixels will be replaced with random picks of good pixels in their neighborhood.

## 18. Archive raw movies

This function allows to archive movie stacks in MRC files with each pixel saved in 4 bits. This function can only be enabled when (1) the input movie stack is given in MRC file, (2) the MRC file has 8 bits per pixel corresponding to MRC mode of 0 or 5, and (3) the full path of the archive file is provided behind "-ArcDir".

If the gain reference is provided as a MRC file on the command line, the gain reference will be saved at the end of the extended-header area in the archive file.

## 19. Taking into account of frame blurring

Since version 1.1.0, there is a newly introduced option "-InFmMotion 1" that takes into account of motion-induced blurring of each frame. The test on T20S proteasome data set shows, although not significant, noticeable resolution improvement of reconstruction. By default, this option is off.

## 20. Generate even and odd sums

Users can use "-SplitSum 1" to generate even and odd sums that are the partial sums of even and odd frames, respectively. The corresponding MRC files are appended with "EVN" and "ODD", respectively.

## 21. On-the-fly motion correction

Since version 1.3.2, MotionCor2 has added a function to facilitate the on-the-fly motion correction. This function allows a new process to be started without being tied to specific GPU(s). As long as there are free GPUs at the moment of start, which are not in use by any other MotionCor2 processes, the process can be started immediately. Once the process finishes, it frees its GPU(s), which then become available for new processes. This function is enabled with -Gpu and -UseGpus where -Gpu lists all the GPU IDs installed on your system and -UseGpus specifies number of free GPUs to be used in the process.

At UCSF we are able to perform real-time motion correction when the data collection runs as fast as 4 K3 movies per minute with each containing 200 frames. This is achieved by running 8 MotionCor2 jobs in Scipion on a Linux workstation equipped with 756 GB CPU memory and 8 nVidia 2080ti cards. Each process is started with

```
-Gpu 0 1 2 3 4 5 6 7 \
-UseGpus 1
```

Please be reminded that the architecture PCIe Bus plays a significant role in the overall performance of running multiple jobs simultaneously since MotionCor2 involves intensive data exchange between CPU and GPU memory. In general, more PCIe bus lanes are strongly preferred. If interested, please feel free to contact the author for more information.

## 23. Miscellaneous

Starting the program without specifying any argument will display all the command line parameters with brief descriptions.

## Release Report

**11-30-2016 version:**

1. Bug fix in generation of log file.
2. Bug fix in saving MRC file. 1) Add min, max, mean in the main header. 2) Revise the main header in accordance to the format defined in JSB 2015, 192 (2) 146-150.
3. Bug fix in the determination of whether the input MRC file is a tomographic tilt series or a single-particle movie stack.

**10-19-2016 version:**

1. Bug fix in generation of log file.
2. Bug fix in saving MRC file. 1) Add min, max, mean in the main header. 2) Revise the main header in accordance to the format defined in JSB 2015, 192 (2) 146-150.
3. Bug fix in the determination of whether the input MRC file is a tomographic tilt series or a single-particle movie stack.

**01-03-2017 version:**

1. Add function to correct anisotropic magnification.
2. Add –FmRef that allows movie stacks to be aligned to a user-specified frame.

### Version 1.0.0 – Release on 07-05-2017

Per user requests, we start using version number instead of date to track the release of MotionCor2. The first version number indicates major changes have been made. The second number denotes minor changes or features added. The last number typically refers to bug fixing.

1. Allow users to specify camera defects in an input file.
2. Support batch processing of TIFF stacks.
3. Support archiving MRC stacks.
4. Revised how users should enter the parameters of anisotropic magnification.
5. Provide "-PhaseOnly" option for cross correlation used in alignment.

### Version 1.0.1 – Release on 09-06-2017

This version significantly improves computational efficiency by buffering as many frames in GPU memory as possible. This strategy significantly reduces the overhead of copying frames from CPU to GPU. As a result, as much as 50% of the computational time can be saved.

Importantly, each GPU cannot be shared by two MotionCor2 processes at the same time. If a GPU is used in one MotionCor2 process, the second MotionCor2 process should not use this GPU. Otherwise, both processes will yield incorrect results and CUDA errors due to GPU memory limitation.

The new functions are:
1. Dark-reference correction.
2. Overlapped patches for local motion correction.

## Version 1.1.0 – Release on 09-06-2017

This version further improves the speed of motion correction and more robust compared to previous releases.

The new functions are:
1. -GpuMemUsage allows users to specify how much GPU memory is used to buffer movie frames.
2. -InFmMotion allows to take into account motion-induced blurring on each frame.
3. -Bft takes an optional second parameter that is used for measuring local motion.

## Frequently Asked Questions

1. The input movie stack is already gain corrected. MotionCor2 reports on the terminal "Apply gain to the stack". Will the gain reference be applied again in MotionCor2?
No, as long as the gain reference is not provided from the command line as a MRC file or not contained in the extended header of the MRC file of the input movie stack.

2. Are the bad and hot pixels detected different from camera_defects_pixels?
Yes, they are different since MotionCor2 detects them dynamically.

3. I got the following error messages, what went wrong?
"Error:  CCufft2D failed, unable to create CUFFT_R2C plan."
"Error: CCufft2D::Forward: an illegal memory access was encountered."
"Error:  CCufft2D::Forward: an illegal memory access was encountered."
These error messages are most likely caused by incompatibility between CUDA driver and CUDA toolkit. Ask system administrator to check if both the driver and the toolkit are of the same version.

4. I noticed that the output and log file always list the frame shifts relative the first frame, no ma matter what value "-FmRef" is set to.
The output and log files list the shifts relative to the first frame. However, the correction is relative to the central frame by default. "-FmRef" is a switch that allows to choose the reference either the central frame by giving it a non-zero value or the first frame by setting it zero.

5. Does MotionCor2 work on Falcon images?
Yes.