frontiers in Neuroinformatics

METHODS
published: 16 February 2016
doi: 10.3389/fninf.2016.00006

moco: Fast Motion Correction for Calcium Imaging

Alexander Dubbs[1]*, *James Guevara*[2] *and Rafael Yuste*[2]*

[1] Departments of Mathematics and CSE, University of Michigan, Ann Arbor, MI, USA, [2] Neurotechnology Center, Department of Biological Sciences, Columbia University, New York, NY, USA

Motion correction is the first step in a pipeline of algorithms to analyze calcium imaging videos and extract biologically relevant information, for example the network structure of the neurons therein. Fast motion correction is especially critical for closed-loop activity triggered stimulation experiments, where accurate detection and targeting of specific cells in necessary. We introduce a novel motion-correction algorithm which uses a Fourier-transform approach, and a combination of judicious downsampling and the accelerated computation of many L_2 norms using dynamic programming and two-dimensional, fft-accelerated convolutions, to enhance its efficiency. Its accuracy is comparable to that of established community-used algorithms, and it is more stable to large translational motions. It is programmed in Java and is compatible with ImageJ.

OCIS codes: Machine Vision Algorithms 150.1135

Keywords: motion correction, calcium imaging, fourier transform, dynamic programming, mesoscale neuroscience

OPEN ACCESS

Edited by:
Qingming Luo,
Huazhong University of Science and Technology-Wuhan National Laboratory for Optoelectronics, China

Reviewed by:
Steve Poelzing,
Virginia Tech, USA
Shaoqun Zeng,
HUST, China

***Correspondence:**
Alexander Dubbs
alex.dubbs@gmail.com;
Rafael Yuste
rmy5@columbia.edu

Received: *21 October 2015*
Accepted: *01 February 2016*
Published: *16 February 2016*

Citation:
Dubbs A, Guevara J and Yuste R (2016) moco: Fast Motion Correction for Calcium Imaging. Front. Neuroinform. 10:6. doi: 10.3389/fninf.2016.00006

1. INTRODUCTION

Calcium imaging, first used to measure the activity of neurons in the early 1990s (Yuste and Katz, 1991), has been successfully applied throughout the nervous system. It allows us to measure the activity of neurons *in vivo*, using either chemical or genetic calcium indicators, with confocal microscopy, two-photon microscopy, or wearable imaging devices (Grienberge and Konnerth, 2012). As a result, it is an increasingly useful tool for identifying the neural circuits underlying behavior. However, calcium imaging videos have challenging noise properties, including white noise and motion artifacts which must be corrected in a preprocessing step before proper analysis can be undertaken.

Motion correction is the first critical step in the analysis of calcium images. After movies are motion-corrected, ROIs are identified, and time-activity graphs are made from each ROI. If the motion-correction is low-quality, then the time-activity graphs suffer, and the reconstructed networks may have errors. If the motion correction is slow, real time closed loop experiments cannot be done while the mouse is in the microscope.

TurboReg (Thevanaz et al., 1998) is a commonly used algorithm for motion correction. It uses a downsampling strategy, a prerequisite for speed, and a template image, necessary for accuracy. We have independently developed a related method, called *moco* (MOtion COrrector), which adopted both strategies, since correcting one image against the next in the stack results in unacceptable roundoff errors. Other approaches use HMMs (Collman, 2010; Kaifosh et al., 2014) or other techniques (Guizar-Sicairos et al., 2008; Li, 2008; Greenberg et al., 2009; Poole et al., 2015; Ringach, unpublished). Guizar-Sicairos et al. (2008) is the only one similar mathematically to, and may be slightly faster than moco, but it has accuracy problems (see **Figures 2**, **3**).

moco uses downsampling and a template image, and it can be called from ImageJ. However, it is faster than TurboReg (Thevanaz et al., 1998) at translation-based motion correction because it uses

dynamic programming and two-dimensional fft-acceleration of two-dimensional convolutions. Guizar-Sicairos et al. (2008) also uses the fft approach with a different objective function that does not require dynamic programming, so our approach is more robust to corrupted data, (see **Figures 2**, **3**). Image Stabilizer is as fast for small images, but is very slow for standard-size images. Running on our own datasets, moco appears faster than all approaches compatible with ImageJ.

moco corrects every image in the video by comparing every possible translation of it with the template image, and chooses the one which minimizes the L_2 norm of the difference between the images in the overlapping region, D, divided by the area of D. The fact that it is so thorough makes it robust to long translations in the data. More complicated non-translation image warps are usually unnecessary for fixing calcium images, which suffer from spurious translations, which moco corrects, and spurious motion in the Z-direction, something difficult to correct. Our approach also uses cache-aware upsampling: when an image is aligned with the template in the downsampled space, it must be jittered when it is upsampled to see which jitter best aligns with the upsampled template. We do this in such a way that data that is used recently is reused immediately, making the implementation faster. Hence, moco is an efficient motion correction of calcium images, and is likely to become a useful tool for processing calcium imaging movies.

TABLE 1 | This table compares the speed of moco to competing algorithms.

Size	moco	TurboReg	TurboReg (slow)	Image Stabilizer
512 × 512 × 1500	66 s	110 s	242 s	304 s
512 × 512 × 2000	90 s	170 s	298 s	464 s
512 × 512 × 6984	288 s	632 s	1303 s	2277 s
416 × 460 × 1000	35 s	71 s	132 s	41 s
256 × 256 × 2028	84 s	121 s	154 s	34 s

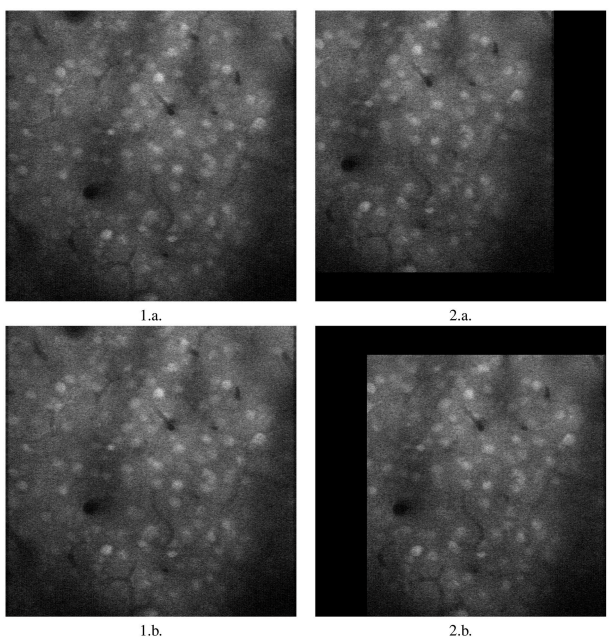


FIGURE 1 | Image registration with moco. 1.a. and 2.a. are first two images of a long, badly corrupted video submitted to moco. 1.b. and 2.b. are the two corrected images. Note that 1.a. and 1.b. are identical, since 1.a. is used as the template image. However, 2.b. is registered by moco, moved to the right to overlap 1.b., it matches it almost perfectly except for the non-overlapping black rim. Images are 317.44 × 317.44 mm.

2. MATHEMATICAL DEVELOPMENT

Let $a_{i,j}$, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ be an image in the stack. Let $b_{i,j}$ be a "template" image against which all other images are aligned, it is typically the first image, a particularly clear

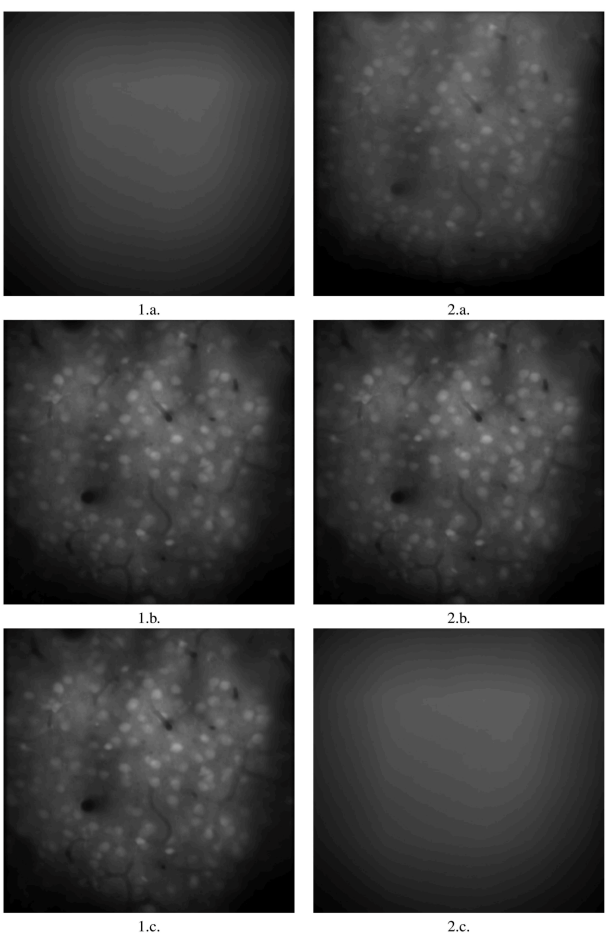


FIGURE 2 | Image registration with two comparable algorithms. 1.a. is the mean of all frames in a badly corrupted video. 2.a. is the of corrected video using our implementation of the (Guizar-Sicairos et al., 2008) approach. 1.b. is the mean of the corrected video using moco. 2.b. is the mean of the corrected video using TurboReg (accurate mode), Thevanaz et al. (1998) 1.c. is the mean of the corrected video using TurboReg (accurate mode). 2.c. is the mean of the corrected video using Image Stabilizer. Note that moco and TurboReg have superior performance, as noted by the sharper and brighter appearance of the cell bodies. Images are 317.44 × 317.44 mm.

image, or the average of the images. We assume a is downsampled if it is larger than 256 × 256. We want to pick (s, t) such that $\max(|s|, |t|) < w$, where w is input by the user, and

$$f_{s,t} = \frac{1}{\text{Area}(D_{s,t})} \sum_{(i,j)\in D_{s,t}} (a_{i+s,j+t} - b_{i,j})^2$$

is minimal, where $D_{s,t}$ is the set of ordered pairs of integers (i', j') such that $1 \le i' \le m$, $1 \le j' \le n$, $1 \le i' + s \le m$, and $1 \le j' + t \le n$. If we do this for every image a in the stack, we have then motion corrected the video, and we are done, up to a short upsampling step. No ROIs (regions of interest) are used, we use the whole image in every frame in the stack. To upsample, multiply the optimal (s, t) by 2 and do a local search to minimize $f_{s,t}$ on the finer grid. Now,

$$\text{Area}(D_{s,t}) f_{s,t} = \sum_{(i,j)\in D_{s,t}} a^2_{i+s,j+t} + \sum_{(i,j)\in D_{s,t}} b^2_{i,j}$$

$$-2 \sum_{(i,j)\in D_{s,t}} a_{i+s,j+t} b_{i,j}.$$

The first two sums can be computed via dynamic programming. Let's consider a when s and t are negative. Let

$$g_{s,t} = \sum_{(i,j)\in D_{s,t}} a^2_{i+s,j+t}.$$

We have that

$$g_{s,t} = g_{s-1,t} + g_{s,t-1} - g_{s-1,t-1} + a^2_{m+s,n+t}.$$

Hence, the first two sums can be computed for all (s, t) in $O(mn)$ time, which is unaffected by a constant amount of downsampling. It suffices to compute for all (s, t) such that $\max(|s|, |t|) < w$,

$$h_{s,t} = \sum_{(i,j)\in D_{s,t}} a_{i+s,j+t} b_{i,j}.$$

Let $\hat{b}$ be b rotated 180 degrees. Using MATLAB notation, let

$$\tilde{a} = \text{fft2}([[a, \text{zeros(m, w)}]; \text{zeros(w, n + w)}]),$$

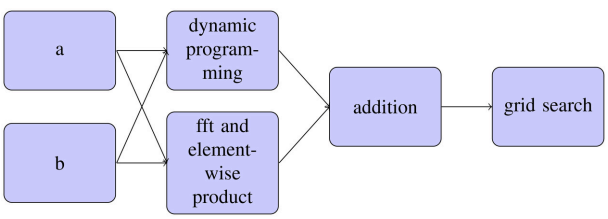
$$\tilde{b} = \text{fft2}([[\hat{b}, \text{zeros(m, w)}]; \text{zeros(w, n + w)}]).$$

Commas denote horizontal concatenation, semicolons denote vertical concatenation, and zeros(x, y) is an $x \times y$ matrix of zeros. For equally sized matrices X, Y, let $Z = X \odot Y$ mean $Z_{i,j} = X_{i,j} Y_{i,j}$. Then

$$\text{ifft2}(\tilde{a} \odot \tilde{b})$$

is a rearrangement of h. Since fft2's are fast, that means h can be computed for all (s, t) in $O(mn \log(mn))$ time. Hence, after upsampling, the entire video can be aligned in $O(mnT \log(mn))$ time, where T is the number of slides in the video. This includes an $O(mn)$ pixel-by-pixel search for the optimum of $f_{s,t}$ over all possible (s, t) pairs.

After (s, t) are chosen to minimize $f_{s,t}$, they are multiplied by two multiple times to upsample. Every time they are multiplied by 2, $f_{2s+u,2t+v}$ are computed for $u, v \in \{0, -1, 1\}$ to see which u and v are minimal. These nine evaluations of f are done with a cache-aware algorithm for speed. The following flowchart describes the algorithm.



3. RESULTS

In **Table 1** we compare moco in speed to TurboReg (Thevanaz et al., 1998) on its translation mode, using both the "fast"

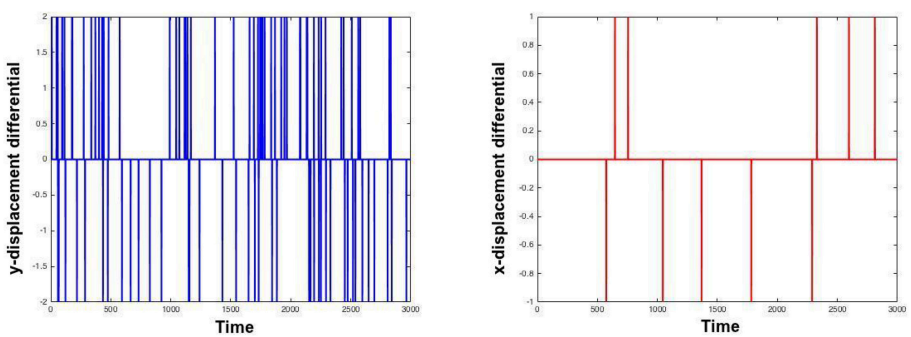


FIGURE 3 | Analysis of spurious translations by moco and a similar algorithm. Left and right images *i* right show differences in displacement in the first and second dimensions as a function of time in moco and (Guizar-Sicairos et al., 2008). The video on which they were applied was a real calcium image with added horizontal and vertical spurious translations, to make the task more difficult. moco and (Guizar-Sicairos et al., 2008) generate different translations, and the differences in the translations found are plotted. The left plot shows the y-translations that moco makes minus the y-translations that (Guizar-Sicairos et al., 2008) makes. This difference is typically zero, but there are notable exceptions. The right plot does the same thing for x-translations. Note how moco detects many more translations.

and "accurate" settings. We also compare it to Image Stabilizer using its default settings (Li, 2008) (it can be made faster by changing the settings but the accuracy is poor). We perform the comparison on several real calcium imaging videos, which we say are $m \times n \times T$ if they contain T slides of size $m \times n$. If the images are larger than 256×256, we downsample once, otherwise, we do not downsample. We have found that dowsampling 3 and 4 times causes severe errors so we avoid those settings. In addition, we have compared moco to TurboReg on synthetic images with severe translational motion artifacts and have found that moco is slightly more accurate. All times are in seconds. The template used for every video is the first image in the video for both moco and TurboReg. moco uses a maximum translation width of $\min(m, n)/3$ in both the i and j directions.

As is clear from the **Table 1**, moco is faster than its most used current method, TurboReg. It may be marginally slower than (Guizar-Sicairos et al., 2008), but **Figures 2**, **3** prove that a code we have created to have similar results to Guizar-Sicairos et al. (2008) is inaccurate. **Figure 1** shows the first two images of a corrupted video on the first row. moco corrections are on the second row. It is clear that moco can fix the image motion, even though the problems with it are severe. **Figure 2** shows the mean image from a corrupted video (i.e., add every image in the stack together via matrix addition and then divide the resulting matrix by the number of images in the stack), and the mean image of moco and TurboReg corrections, as well as the correction from our MATLAB version of the (Guizar-Sicairos et al., 2008) algorithm. Note that the (Guizar-Sicairos et al., 2008) algorithm artificially fades the image, indicative of poor alignment, whereas our algorithm and TurboReg are crisp, indicating good alignment. **Figure 3** shows the differences in x and y displacement done by moco and the (Guizar-Sicairos et al., 2008) algorithm in attempt to correct a corrupted calcium imaging video with added spurious translations.

AUTHOR CONTRIBUTIONS

AD is the first author, RY is the supervisor and corresponding author, JG is a programmer.

ACKNOWLEDGMENTS

We would like to thank Julia Sable for her help, and Inbal Ayzenshtat, Jesse Jackson, Jae-eun Miller, Luis Reid, Weijian Yang, and Weiqun Fang for their datasets. The first author would also like to thank the Columbia University Data Science Institute for their ongoing support. This research is supported by the NEI (DP1EY024503, R01EY011787), NIHM (R01MH101218) and a DARPA contract, SIMPLEX N66001-15-C-4032. This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U.S. Army Research Office under contract number W911NF-12-1-0594 (MURI).

REFERENCES

Collman, F. (2010). *High Resolution Imaging in Awake Behaving Mice: Motion Correction and Virtual Reality*. Ph.D. thesis, Princeton University.

Greenberg, D.S., and Kerr, J. N. (2009). Automated correction of fast motion artifacts for two-photon imaging of awake animals. *J. Neurosci. Methods* 176, 1–15. doi: 10.1016/j.jneumeth.2008.08.020

Grienberge, C., and Konnerth, A. (2012). Imaging calcium in neurons. *Neuron* 73, 862–885. doi: 10.1016/j.neuron.2012.02.011

Guizar-Sicairos, M., Thurman, S. T., and Fienup, J. R. (2008). Efficient subpixel image registration algorithms. *Optics Lett.* 33, 156–158. doi: 10.1364/OL.33.000156

Kaifosh, P., Zaremba, J. D., Danielson, N. B., Losonczy, A. (2014). SIMA: Python software for analysis of dynamic fluorescence imaging data. Front. Neuroinform. 8:80. doi: 10.3389/fninf.2014.00080

Li, K. (2008). *The Image Stabilizer Plugin for ImageJ*. Available online at: http://www.cs.cmu.edu/~kangli/code/Image_Stabilizer.html

Poole, B., Grosenick, L., Broxton, M., Deisseroth, K., and Ganguli, S. (2015). "Robust non-rigid alignment of volumetric calcium imaging data," in *COSYNE*.

Thevanaz, P., Ruttimann, U. E., and Unser, M. (1998). A pyramid approach to subpixel registration based on intensity. *IEEE Trans. Image Process.* 7, 27–41. doi: 10.1109/83.650848

Yuste, R., and Katz, L. C., (1991). Control of postsynaptic Ca^{2+} influx in developing neocortex by excitatory and inhibitory neurotransmitters. *Neuron* 6, 333–344. doi: 10.1016/0896-6273(91)90243-S

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.