ORACA – an open-source software for Online Real-time neural Activity extraction and offline Cross-session Analysis

Weihao Sheng[1], Xueyang Zhao[1], Yang Yang[1*]

1. School of Life Science and Technology, ShanghaiTech University, Shanghai 201210, China.

* correspondence: yangyang2@shanghaitech.edu.cn

Abstract

The idea to combine i*n vivo* functional imaging with optogenetic stimulation to achieve closed-loop, all-optical recording and manipulation of neurons and neural circuits is appealing yet challenging. Other than necessary hardwares, it requires an analysis software fast enough to extract neural activities from imaging data in real time. Here we present an open-source, integrative image processing toolbox ORACA (Online Real-time Activity extraction and offline Cross-session Analysis), which provides solutions for both fast online and accurate offline data analyses. We developed fast GPU-based algorithms that can finish raw image registration, automatic identification and activity extraction of neurons within seconds after image acquisition. Offline analysis pipeline features a new cross-session alignment algorithm that takes into account the angle differences across imaging sessions, useful for shared microscope or long imaging intervals. A modular, user-friendly software that can be used as a complete package or independent modules, ORACA can effectively facilitate the image analysis process, especially for all-optical closed-loop control and long-term repeated imaging.

Introduction

Closed-loop control is a key concept in control theory, in which prompt feedback is incorporated in the workflow to achieve dynamic modulation of the whole system. In neuroscience, instant control of neurons based on real-time activities can be used to monitor response patterns in a more realistic setting, validate models, or treat diseases such as epilepsy [1]. With the development of genetically encoded fluorescent sensors [2-5], *in vivo* microscopy [6, 7], optogenetics [8-10], and holographic optical stimulation [11-13], it is theoretically possible to monitor and manipulate the activity of neurons and synapses using a closed-loop, all-optical method. Such method allows targeting of specific cell types for both observation and manipulation, and the multiplexing feature of light enables concurrent control of different neuronal population using discrete wavelengths. Nowadays, the imaging and stimulation hardware and molecular

tools that are needed for all-optical closed-loop control are already available. But to close the gap, a real-time image analysis software that is capable of identifying active neurons from raw imaging data online is needed. To achieve this, new ultra-fast image registration and neural activity extraction algorithms need to be developed.

Long-term *in vivo* imaging of the same neurons and synapses repeatedly over a prolonged period is routinely used to investigate the dynamics and plasticity of neural circuits [14-17]. During this process, large-scale imaging data are generated**,** calling for efficient and accurate image processing methods. For comparative analysis of repeated imaging at different times or under different conditions, it is especially important to combine datasets from multiple imaging sessions and identify the same regions of interest (ROI), such as neurons and subcellular structures.

There are already several toolboxes available for processing calcium imaging data. Suite2p [18] is a Python package that uses phase correlation for registration and matrix factorisation methods for active component extraction with affordable computational power. CaImAn [19] detects active neurons using a convolutional neural network (CNN), showing improved performance for real-time streaming data. Mitani and colleagues developed a processing pipeline including fast motion registration for closed-loop experiments [20], using OpenCV libraries for registration and a kernel density method for baseline estimation. All these softwares provide solutions for extracting neuronal calcium activity from imaging datasets, but there is plenty of room for improvement in processing speed and accuracy. Currently, there is no software suitable for online extraction of active neurons or cellular components on a trial-by-trial basis. Moreover, for offline analysis of repeated imaging of the same cell population, although CaImAn incorporated a cross-session mask alignment option, it did not take into account the inter-session frame distortion caused by imaging angle and depth differences, which is often inevitable for shared microscopes or long-term imaging spanning weeks or months. In addition, adequate quality control steps are lacking [21]. Thus we developed ORACA (online Real-time Activity extraction and offline Cross-session imaging Analysis), which includes an improved registration algorithm much faster than other popular methods, a fast trial-by-trial activity extraction algorithm as well as a correlation-based, optimized method for accurate offline active cell recognition, and a cross-session alignment algorithm that can detect angle differences across imaging sessions to ensure accurate tracking of the same neurons. Altogether, ORACA can markedly accelerate processing of imaging data, both in online and offline settings. With data visualization and quality check options readily available, ORACA is a ready-to-use, user-friendly software requiring little user intervention and minimal programming ability. All modules are written in MATLAB, and each module can be used individually,integrated in other systems, or as a complete pipeline.

Results

Procedure at a glance

The image processing toolbox consists of two pipelines designed for different purposes: an online real-time analysis pipeline for closed-loop experiments, and an offline multi-session analysis pipeline for accurately extracting and tracking neurons across different imaging sessions. Both pipelines start with "*Image Registration*", in which we incorporated an improved fast registration algorithm to correct for brain pulsation and

movement. Quality check in this step allows users to view the summary image of registered frames and X-Y offset of each frame calculated by the algorithm, based on which users can decide whether additional registration step is needed. For online "*Mask Segmentation*" and "*Activity Extraction*", regions-of-interest (ROIs), or active components, are identified and updated on a trial-by-trial basis, taking into account both the baseline fluorescence and activity level of each component. The fluorescent signal of each ROI is plotted and updated in real time. For offline "*Mask Segmentation*", ROIs were identified automatically based on spatial and temporal features from each registered video. A mask file summarising all ROIs in a given video is produced, and quality check for this step is available with graphical interfaces. Users can inspect the mask file, and if needed, manually add ROIs or delete false-positive ROIs. "*Cross session alignment*" incorporates mask files generated from all imaging sessions. Alignments between sessions are calculated based on average image of registered videos from "*Image Registration*" and segmented masks from *"Mask Segmentation"*, with the possibility to correct for imaging angle differences. Then ROIs from different imaging sessions are merged, grouped and indexed. The key in this step is to correctly identify and link the same ROIs across different sessions, for which quality control is essential. Users can manually refine the alignments, and check the indices of the ROIs. "*Activity Extraction*" extracts the fluorescent signal of each ROI, and plots a temporal trace depicting the fluorescent changes of the ROI. We provide options for standard statistical analyses, and data files can also be exported for further analyses.

All the modules have clearly defined inputs and outputs, and each can operate independently. To automate the whole process, they are also packed into a single pipeline, with a user-friendly GUI. As a result, minimal programming knowledge is required to use the package.

Step 1: Image registration efficiently stabilized image frames

Motion artefacts are inevitable for *in vivo* microscopy. Relative motion between the brain and the implanted imaging window, caused by animals' body or limb movements, breathing and heartbeats, is often more pronounced in awake behaving animals, even if they are head-fixed. Removing motion artefacts is essential for subsequent image processing because lateral shift in videos can introduce noise for ROI identification and extraction. To address this issue, many different registration algorithms have been proposed. TurboReg [22] is a commonly used algorithm to correct motion artefacts by minimizing summation of intensity difference between frames. Algorithms such as single-step DFT [23] and moco [24] use discrete Fourier transform to maximize cross-correlation between each frame and a template image. Because we mainly deal with images recorded at high frame rates with low in-frame distortion, we combine several computational methods to achieve fast and accurate registration using translational transformation approach (see Methods for more detail).

To evaluate the registration speed and accuracy of ORACA, we compared it to some popular methods including single-step DFT, TurboReg, and moco, usinga calcium video of dendrites. The example video contains 5040 frames of 512×512 image, with both lateral shifts and a slow directional drift in the x-y plane. ORACA outperformed all other competitors in registration speed, and can be further accelerated ~4 times with GPU optimisation (Figure 2a). The registration accuracy was not compromised, as is shown in the projected image (Figure 2b). From the calculated x-y offset of each frame,

it is clear that ORACA captured both small, fast lateral shifts and large, slow drifts spanning thousands of frames (Figure 2c).

Step 2: Active ROI identification

2.1: Online trial-by-trial segmentation and real-time demonstration of active components

Identification of active cells or subcellular components is critical for automated image processing, and identification speed is especially crucial for closed-loop experiments. Matrix factorisation methods (Pnevmatikakis et al., 2017) or graph cut algorithms (Speen et al., 2019) have been used to segment pixels into active and inactive clusters. These methods require considerable computational power and a massive amount of imaging data for effective identification of active components, thus not suitable for trial-by-trial, real-time analysis of a small number of images.

To identify active neurons or subcellular components in real time, we developed a variance scaling method based on area under curve (AUC) and standard deviation of videos obtained in one single trial, typically containing a few tens or hundreds of images (see Methods for more details). Components that show activity or high baseline fluorescent levels are effectively segmented, and regions that exceed a user-defined threshold are labelled as regions of interest (ROIs) (Figure 3a). Activity traces of all identified areas are calculated instantly and plotted within seconds after image acquisition (Figure 3b).

2.2: Mask segmentation automatically identifies cells as ROIs

We incorporated a published algorithm, HNCcorr, for offline automatic identification of active components in registered videos [25]. In an example field of view (FOV) from one cell recording session (Figure 4a,b), HNCcorr successfully identified multiple active components in the session (Figure 4c). Most segmented components overlap with the projected image (shown in yellow in Figure 4c), yet HNCcorr effectively identified several components unidentifiable in the projected image based on their rare but strong activities (shown in green in Figure 4c). Pixels identified as belonging to one cluster show similar activities compared to an outside pixel (Figure 4d & 4e).

For quality control, we introduced a user-friendly graphical interface to examine clustering results by automated segmentation. Since HNCcorr cannot detect cells that showed no change in fluorescence even if they are fluorescently labelled, users can choose to manually add these cells. Users can also manually exclude falsely identified cells or refine identified segments.

Step 3: Cross session alignment corrects for inconsistent imaging angles and merges ROIs from multiple imaging sessions

Accurate inter-session alignment is a pre-requisite for tracking the same ROIs across imaging sessions. Inter-session variations can originate from the microscope (e.g. shift in x/y/z planes, angle of objective lens, microscope parameter settings) and the animal (relative movements between the brain tissue and the implant; angle of fixation; tissue growth leading to degraded image quality). Nowadays, an increasing number of microscope models have flexible imaging angles, achieved by rotation of the objective lens or the whole light path. This feature made it easier to image brain areas such as auditory cortex, which is on the side, but also was more likely to cause inconsistency of imaging angles in different sessions. The problem is very common since in most laboratories or facilities, an expensive device such as a multi-photon microscope is shared by many experimenters, and the microscopes are constantly adjusted to meet each experimenter's needs.

We implemented an affine transformation method in our cross-session alignment step to solve this problem, which can correct for the angle differences between sessions. When imaging sessions were taken on the same z plane with the same angle, inter-session variations can be easily corrected by rigid transformation [19, 26]. But if two sessions were imaged from different angles, an affine transformation is needed to partially correct for such variations, in order to gain a maximal likelihood of the FOVs from these sessions (Figure 5a,b; see Methods for more details). As shown in Figure 5b, our method successfully aligned two sessions, where cells from two sessions showed overlaps both in the centre and at margins (indicated in yellow and white arrows). All sessions were subsequently aligned using the same method (Figure 5c), while masks from all sessions were merged into one unified mask which was later labelled with corresponding indices and applied to all sessions (Figure 5d). We use a 'capture-all' stratery that keeps all identified components from all sessions, so any component that ever showed neural activity are included, even if they are only active in one of many imaging sessions. (Figure 5e & 5f). In case of significant shift in the x/y plane, some components are cropped out after registration and alignment.

Users can assess the quality of cross-session alignment by evaluating an overlay image of z-projections of any two (or more) imaging sessions. The alignment can be refined manually, or using other user-defined algorithms

Step 4: Activity extraction and neural signal demonstration of ROIs cross multiple sessions

After indexing all ROIs for all imaging sessions, we can extract and compare the neural responses across different sessions. We calculate changes in fluorescent signals ($\Delta F/F$) in the indexed ROIs. $\Delta F/F$ for each ROI is calculated by averaging activities of all pixels in the ROI (see Methods for more detail). Traces of the same ROI across all sessions are grouped, sorted and plotted. We first demonstrate an example neuron that showed activity in three sessions (Figure 6a & 6b). The cell exhibited reliable stimulus-evoked responses to the 21.4kHz pure tone, different responses to the

15.4kHz pure tone (Methods, Figure 6b), and was unresponsive to lower frequencies. In comparison, another example neuron was responsive to 5.6kHz only in one session (Figure 6c & 6d), while staying silent in other times. The latter cell was cropped out in the third session, showing a straight line and NaN (not-a-number) in the plot (Figure 6d, last line), but was kept in the master mask.

Discussion

We aim to develop a multifunctional analysis software that can deal with a small amount of imaging data fast enough for closed-loop stimulations, and a large dataset of multiple repeated imaging sessions with high speed and accuracy. The ORACA system combines multiple state-of-the-art algorithms to achieve unprecedented speed for automatically processing calcium imaging data, takes as input the raw imaging videos of a single trial or multiple imaging sessions, automatically detects active components at near-human performance, and outputs activity traces of neurons or subcellular structures for each trial or for all imaging sessions. The whole analysis procedure requires minimal user intervention and programming ability.

The GPU-based registration algorithm is much faster than most other registration algorithms: it took less than one second to register 60 frames of 512×512 image for online real-time processing, and only 20 seconds for over 5000 frames in a complete imaging session (Figure 2a). To our knowledge, only one method using an external OpenCV package written in C++ showed comparable registration speed [20]. When it comes to parametres, for our registration module to run effectively, it is important to choose a reasonable number for the maximum shift, ms. A ms too large (e.g., for a 512*512 image, ms set as 170) will possibly result in faulty matches, while a ms too small (e.g., for a 512*512 image, ms = 1) will cause the registration to fail because the x-y shift between frames is likely to exceed 1 pixel. It is advised that users start with the default setting and adjust from there.

The development of holographic optical stimulation combined with optogenetics makes it possible to manipulate the activity of multiple synapses, neurons or neuronal ensembles in a temporally and spatially precise manner. This technology will be especially powerful if the activity of the neurons can be monitored in real time and hence manipulated in an all-optical closed-loop fashion. The bottleneck often lies in the analysis of real-time imaging data. In a behaving animal, the image registration, active component identification and activity extraction of the current imaging data need to be fast enough to be incorporated into a closed-loop control experiment. For a behavioral experiment to run smoothly, be it a motor learning task, decision making or classical conditioning, the whole image processing should take no more than a few seconds. In our experimental setting, each trial consists of 60 frames (512×512 px in size) of recorded neuronal responses to an auditory stimulus. Our online analysis pipeline finishes the whole process and plots the active component contours and response curves within 5 seconds after image acquisition, within the range of a common inter-trial interval. It is thus possible to perform a closed-loop stimulation experiment on a trial-by-trial basis, without interrupting a normal behavioral training procedure.

HNCcorr is an algorithm that automatically identifies active cell-like components in a

given image series of recorded calcium signal [25]. They treat cells as clusters of pixels, and identify cells using the clustering problem Hochbaum's Normalized Cut (HNC), in which they also devised a novel similarity measure named $(SIM)^2$ ("similarity squared"). This integrated approach outperformed most of other algorithms that do not rely on machine learning methods for identification on the Neurofinder benchmark [27]. We modified this technique to identify active cells in image series (Figure 4), and with some modifications it can be expanded to identify spines and boutons alike. The original HNCcorr algorithm is a correlation-based approach and relies on the activity of the clustered pixels, thus it lacks the ability to extract spatial information. This shortcoming is more prominent when it will be expanded to spine/bouton identification, because the coordinated signals of active dendrites become a major source of interference. Therefore, manual inspection of the identified components is still required to exclude false-positives and add unidentified components.

For cross-session alignment, we are the first to use spatial features of the original FOV to align different sessions with possible imaging angle offsets. The ability to correct for imaging angle inconsistency is the biggest difference between out method and two previous studies which provided solutions for cross-session alignment, RegisterMulti [19] and CellReg [26]: RegisterMulti uses the Hungarian algorithm to relate two components together while keeping the contour of identified components untouched, which requires extra computation; CellReg aligned segmented masks and applied a probabilistic model using centroid distance, an arbitrary parameter that may needs.
We use average image of each session as spatial features to properly align two sessions and generate transformation matrix for alignments. At the core of our method lies the conception that variations between sessions originate from angle differences between optical lens and the imaging window, and that later sessions can be seen as different projections onto the first one. Therefore, we assign the first session as the standard FOV, and calculate affine displacements between later FOVs and the first FOV, thus correct for angle differences. The two other algorithms assumed no imaging angle differences across sessions, and therefore used non-rigid registration, allowing only for translational and rotational corrections. Using our affine-based approach, angle difference up to 5° can be corrected while maintaining most identified ROIs even if they reside on the edge. Note that angle differences too large may result in loss of marginal components, and this cannot be fixed in post-processing pipelines.

This part of the software can also be used for real-time quality check as well as calibration during image acquisition: a fast calculation of the imaging angle can inform the experimenter of the angle offset, and if it is off by over 5°, it is advised that the experimenter re-adjust the microscope to avoid loss of data. As more microscope models allow flexible adjustment of objective lens angles, our method is more applicable for these systems. Before the actual experiment, users can record a small number of images and calculate the angle offset from previous sessions, and adjust the microscope accordingly to match previous recordings and thus improve multi-session data quality.

Overall, we designed a versatile imaging analysis toolbox which not only made real-time analysis for Calcium imaging and other fluorescent sensors possible, but also provides solution for accurate offline analysis for multi-session data. Quality check options are included for every step of the analysis pipeline, making ORACA an especially user-friendly toolbox.

Materials and methods

Hardware for post-processing

We used a Dell workstation to do all computations. Our workstation is equipped with two Intel Xeon Gold 5122 CPU, 64GB DDR4 RAM, and one Nvidia Quadro P6000 GPU, running Windows 10 Workstation Edition. TurboReg and moco ran in Fiji (ImageJ 1.48v, Java 1.6.0_24 (64bit)), and single-step DFT and our package ran on MATLAB R2017b (Update 9).

Our package fully utilises the great computational power of GPU to accelerate the processing pipeline. All our modules provide both CPU and GPU versions, and they do mostly the same thing. We recommend using GPU version for best performance.

Preparations of performance test and simulated data

When evaluating registration speed performance, we ran single-step DFT and our approach in MATLAB, TurboReg and moco in Fiji. File input/output operation time was excluded from timing for better evaluation of all algorithms in terms of computation efficiency. Since moco by default did not provide macro access in Fiji, we used a screen recording software to record its registration process and calculated its running time by counting frames in the recorded video when it is registering frames, so timing of moco might be a bit less accurate than others'. Moreover, algorithms running on different platforms may have different running time due to programming language-specific implementations.

Raw image registration algorithms

Motion correction is the first step in calcium imaging analysis pipeline for further analyses. Drifts in the imaging plane can be corrected by finding the maximum similarity between a template image and each frame, and to our knowledge there is currently no algorithm that compensates drifts in z axis (motion in imaging depth). Correction on each frame can be performed by calculating offsets in the current plane and shifting the frame to match the template image.

Inspired by the moco algorithm [24], we combined multiple computational ideas to produce a faster implementation. Let a be a frame in a recorded image series of h rows(height), w columns(width), and $a_{i,j}$ be a pixel of i row, j column in that frame. Under the same notation, we define t to be a template image, which usually is the average of all frames or the first frame in an image series. Images t and a are standardised by mean-centring before registration:

$$I = \frac{I - \bar{I}}{\mathrm{std}(I)}$$

Let ms be the user-defined maximum shift value, and by default it is set to 1/5 of the frame size. We want to find a pair of (x, y) offsets within maximum shifts, so that pixels in the overlapping area between a and t are of minimum difference. This minimum can be found when

$$D(x,y) = \frac{1}{(h-|x|)(w-|y|)} \sum_{overlap} (t_{i,j} - a_{i+x,j+y})^2$$

is minimal, where $(x,y) \in [-ms, ms] \cap \mathbb{Z}$. We substitute the denomiator with A indicating the area of overlapping between the template and each frame, and we rewrite the equation into:

$$A \cdot D(x,y) = \sum_{overlap} t^2 + \sum_{overlap} a^2 - 2 \sum_{overlap} a \cdot t$$

In this equation, $\sum_{overlap} t^2$ is the sum of all pixels squared in the overlapping area of the template. Given that $(x,y) \in [-ms, ms] \cap \mathbb{Z}$, there are finite possibilities of this value, and they can be calculated for any given (x,y) in the range beforehand. The dynamic part in the equation, $\sum_{overlap} a^2$ and $\sum_{overlap} a \cdot t$, are both related to the current frame a. $\sum_{overlap} a \cdot t$ can be seen as the convolution of t over a. Using $\tilde{t}$ as t rotated 180 degrees, we have

$$\sum_{overlap} a \cdot t = \text{conv2}\,(a, \tilde{t})$$

Mathematically, convolution can be efficiently calculated using two-dimensional fast Fourier transform (2D-FFT):

$$\text{conv2}\,(a, \tilde{t}) = \text{ifft2}\,(\text{fft2}\,(a) \cdot \text{fft2}\,(\tilde{t}))$$

and $\sum_{overlap} a^2$ can also be calculated as

$$\sum_{overlap} a^2 = \text{conv2}\,(a^2, \text{ones}(h-2ms, w-2ms))$$

where $\text{ones}(h-2ms, w-2ms))$ is a matrix of size $(h-2ms, w-2ms)$ filled with 1 using MATLAB notation. To efficiently calculate these values, we register template t onto current frame a, so that variables regarding t do not have to be recalculated each time.

We developed a highly optimised `conv2` function (named `conv2_fftvalid`) just for our registration. Our code is way faster than MATLAB's default implementation, and the whole registration process can be further accelerated by downsampling and doing a local search after upsampling. Running on GPU, our code is way faster than most of its competitor algorithms. Pseudo code to illustrate the registration process is listed in Box 1.

Online trial-by-trial identification of active components

To meet the speed requirement for closed-loop imaging and stimulation experiments, we combined several statistical tools to produce fast segmentation of pixels on a trial-by-trial basis. A block is defined as a temporal part of the full video that extends for one trial. The block is first normalised by subtracting mean value and divided by the standard deviation of the whole block:

$$block = \frac{block - \overline{block}}{\text{std}(block)}$$

Then, AUC of all pixels are calculated by summing up all values in their temporal trace, and activities smaller than zero are discarded. To increase the signal-to-noise ratio, we take the standard deviation of the normalised block as a scaling factor. The product matrix undergoes a simple filter that converts it to binary image using a user-defined

threshold (0.05 by default), and eliminates areas smaller than user-defined size value (16 pixels by default). The whole process is illustrated using pseudo-code shown in Box 2, and a demo pseudo-code for fast processing is showin in Box 3.

Offline segmentation of cell-like components using HNCcorr

HNCcorr is an algorithm that automatically identifies active cell-like components in a given image series of recorded calcium signal [25]. HNCcorr requires “seeds”, pixels that have the highest correlation coefficient with their neighbouring pixels, to generate “patches” in the recorded images, on which all subsequent segmentations are performed. For cell identification, seeds are defined as the top 40% pixels that have the highest correlation coefficient with neighbouring pixels in default settings. After initial seeds are generated, HNCcorr takes patches around every seed pixel in the image series as the input, and outputs clustering results that tells how pixels in the patches are segmented into different clusters. Typically, two clusters are given, one representing active pixels and one representing background. An active cell is a cluster of connected pixels defined within contours of the output.

We incorporated the MATLAB version of HNCcorr to accommodate our requirements, while we wrote a simple wrapper function for simultaneous interaction with HNCcorr.

Offline manual segmentation of components in a video

Since most automated identification methods may either falsely identify unwanted components or neglect true active components, we provide an easy-to-use user interface for manual labelling of active components in a given video, incorporated in the `mask_segmentation` module. Our manual labelling program allows users to play the video for full length or just one trial, while adding or erasing segments at the same time.

Cross session alignment algorithms

Variance in FOVs caused by rotational, translational or tilted offsets can be corrected properly by considering different imaging sessions as different three-dimensional projections onto two-dimensional planes, and each imaging plane is a slice taken from different angles. To align two sessions, we consider the second session as projections from a different angle, and apply different transformation methods for different situations.

We collect summary images of every session, and uses affine transformation to properly align different sessions. Refining is also possible by manually select matching points between sessions, or use other user-defined algorithms. For off-plane alignments, we calculate the maximum similarity for affine projection, and take the z-axis angle out of the transformation matrix. Steps of execution are summarised in the form of pseudo code in Box 4.

Source extraction and neuropil contamination removal

For activity extraction, we first calculate the average activity of all pixels in the same

ROI, extracting F_a. $\Delta F/F$ is subsequently calculated as

$$\frac{\Delta F}{F} = \frac{F_{real} - F_0}{F_0}$$

where F_0 is an averaged value of activities in certain time window before stimulus onset. This is known as trial-by-trial calculation of activity.

References

1. Krook-Magnuson, E., et al., *On-demand optogenetic control of spontaneous seizures in temporal lobe epilepsy.* Nat Commun, 2013. **4**: p. 1376.
2. Chen, T.W., et al., *Ultrasensitive fluorescent proteins for imaging neuronal activity.* Nature, 2013. **499**(7458): p. 295-+.
3. Feng, J., et al., *A Genetically Encoded Fluorescent Sensor for Rapid and Specific In Vivo Detection of Norepinephrine.* Neuron, 2019. **102**(4): p. 745-+.
4. Sun, F.M., et al., *A Genetically Encoded Fluorescent Sensor Enables Rapid and Specific Detection of Dopamine in Flies, Fish, and Mice.* Cell, 2018. **174**(2): p. 481-+.
5. Tian, L., et al., *Imaging neural activity in worms, flies and mice with improved GCaMP calcium indicators.* Nature Methods, 2009. **6**(12): p. 875-U113.
6. Svoboda, K., et al., *In vivo dendritic calcium dynamics in neocortical pyramidal neurons.* Nature, 1997. **385**(6612): p. 161-165.
7. Jung, J.C., et al., *In vivo mammalian brain Imaging using one- and two-photon fluorescence microendoscopy.* Journal of Neurophysiology, 2004. **92**(5): p. 3121-3133.
8. Boyden, E.S., et al., *Millisecond-timescale, genetically targeted optical control of neural activity.* Nature Neuroscience, 2005. **8**(9): p. 1263-1268.
9. Deisseroth, K., et al., *Next-generation optical technologies for illuminating genetically targeted brain circuits.* Journal of Neuroscience, 2006. **26**(41): p. 10380-10386.
10. Packer, A.M., et al., *Two-photon optogenetics of dendritic spines and neural circuits.* Nature Methods, 2012. **9**(12): p. 1202-U103.
11. Lutz, C., et al., *Holographic photolysis of caged neurotransmitters.* Nature Methods, 2008. **5**(9): p. 821-827.
12. Reutsky-Gefen, I., et al., *Holographic optogenetic stimulation of patterned neuronal activity for vision restoration.* Nature Communications, 2013. **4**.
13. Yang, W.J. and R. Yuste, *Holographic imaging and photostimulation of neural activity.* Current Opinion in Neurobiology, 2018. **50**: p. 211-221.
14. Driscoll, L.N., et al., *Dynamic Reorganization of Neuronal Activity Patterns in Parietal Cortex.* Cell, 2017. **170**(5): p. 986-+.
15. Peters, A.J., et al., *Reorganization of corticospinal output during motor learning.* Nature Neuroscience, 2017. **20**(8): p. 1133-+.
16. Xin, Y., et al., *Sensory-to-Category Transformation via Dynamic Reorganization of Ensemble Structures in Mouse Auditory Cortex.* Neuron, 2019. **103**(5): p. 909-+.
17. Holtmaat, A., et al., *Long-term, high-resolution imaging in the mouse neocortex through a chronic cranial window.* Nature Protocols, 2009. **4**(8): p. 1128-1144.
18. Pachitariu, M., et al., *Suite2p: beyond 10,000 neurons with standard two-photon microscopy.* 2017: p. 061507.
19. Giovannucci, A., et al., *CaImAn an open source tool for scalable calcium imaging data analysis.* Elife, 2019. **8**.
20. Mitani, A. and T. Komiyama, *Real-Time Processing of Two-Photon Calcium Imaging Data Including Lateral Motion Artifact Correction.* Frontiers in Neuroinformatics, 2018. **12**.

21. Stringer, C. and M. Pachitariu, *Computational processing of neural recordings from calcium imaging data.* Current Opinion in Neurobiology, 2019. **55**: p. 22-31.
22. Thevenaz, P., U.E. Ruttimann, and M. Unser, *A pyramid approach to subpixel registration based on intensity.* Ieee Transactions on Image Processing, 1998. **7**(1): p. 27-41.
23. Guizar-Sicairos, M., S.T. Thurman, and J.R. Fienup, *Efficient subpixel image registration algorithms.* Optics Letters, 2008. **33**(2): p. 156-158.
24. Dubbs, A., J. Guevara, and R. Yuste, *moco: Fast Motion Correction for Calcium Imaging.* Front Neuroinform, 2016. **10**: p. 6.
25. Speen, Q., et al., *HNCcorr: A Novel Combinatorial Approach for Cell Identification in Calcium-Imaging Movies.* Eneuro, 2019. **6**(2).
26. Sheintuch, L., et al., *Tracking the Same Neurons across Multiple Days in Ca2+ Imaging Data.* Cell Reports, 2017. **21**(4): p. 1102-1115.
27. CodeNeuro. *The neurofinder challenge*. 2020; Available from: http://neurofinder.codeneuro.org/.

Figure 1: The overall processing pipelines for both online(left) and offline(right) analysis. Both pipelines consist of three (online) or four (offline) steps, sharing both *Image Registration* and *Activity Extraction* part, and operate differently in *Mask Segmentation* as well as *Cross Session Alignment*. For online analysis, images taken from one trial is loaded, and the full pipeline can be executed in seconds using GPU-based acceleration, showing significant activities that goes beyond defined thresholds. For offline analysis, images taken in one session or multiple sessions can be processed in batch, and quality control is available at all steps (not shown in the figure).

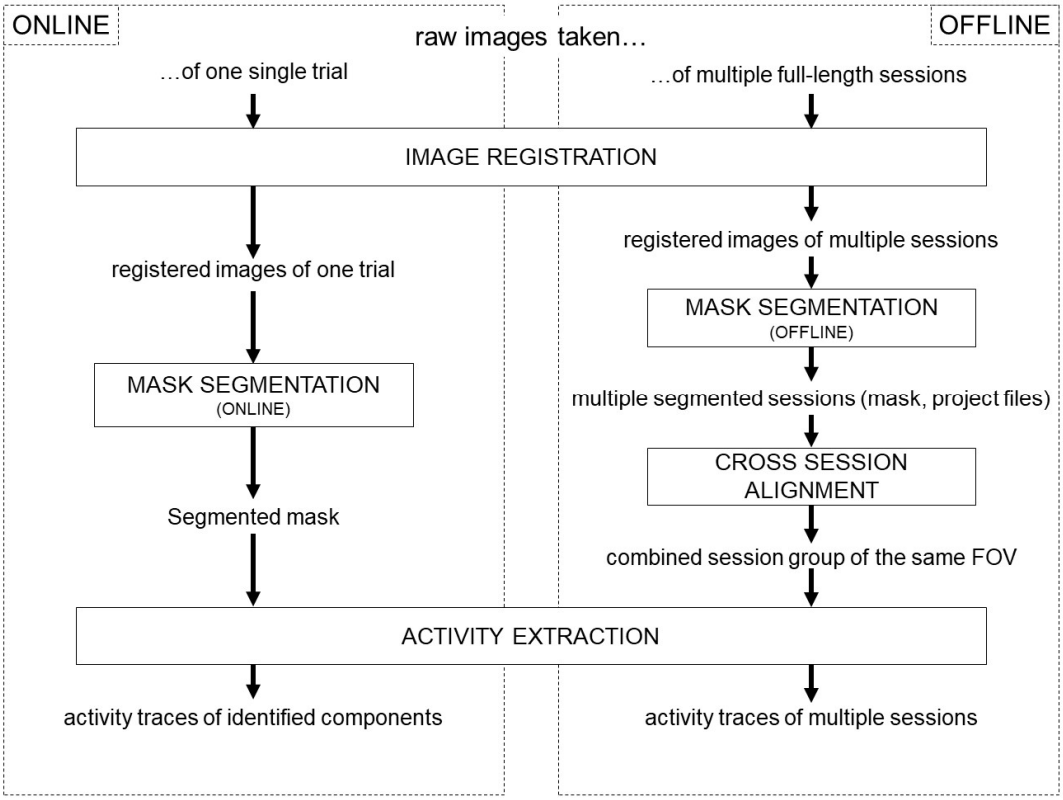


Figure 2: Registration of one example video using different methods. The example video was taken at 15Hz, 512×512 pixels for 5040 frames. Manual inspection of the video revealed that it contained both high -frequency lateral shifts as well as slow drifts into one direction on the X-Y plane. **(a)** Comparison between different algorithms in terms of registration time. Algorithms were timed in its default contexts: TurboReg and moco were timed in Fiji, and other three algorithms were timed in MATLAB. **(b)** Summary images and zommed-in view of original and registered videos. **(c)** Shifts in x-y directions calculated by four algorithms.

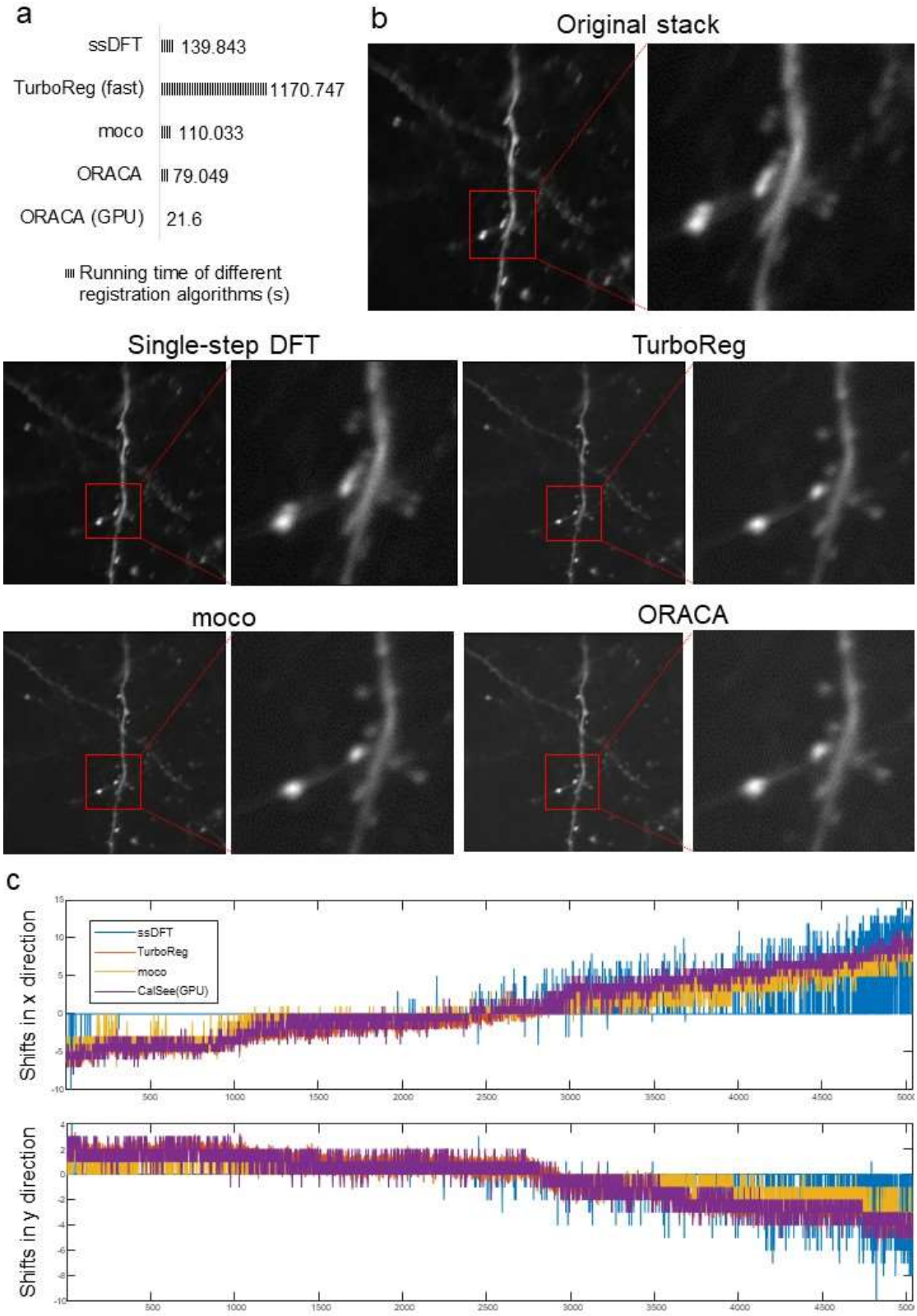

a
ssDFT
139.843
TurboReg (fast)
1170.747
moco
110.033
ORACA
79.049
ORACA (GPU)
21.6
Running time of different registration algorithms (s)
b
Original stack
Single-step DFT
TurboReg
moco
ORACA
c
ssDFT
TurboReg
moco
CalSee(GPU)
Shifts in x direction
Shifts in y direction

Figure 3: Demonstration of Online ORACA user interface. Identified active components in a given trial were (a) labelled with numbers, and (b) their $\frac{\Delta F}{F}$ activities were calculated and sorted from maximum to minimum.

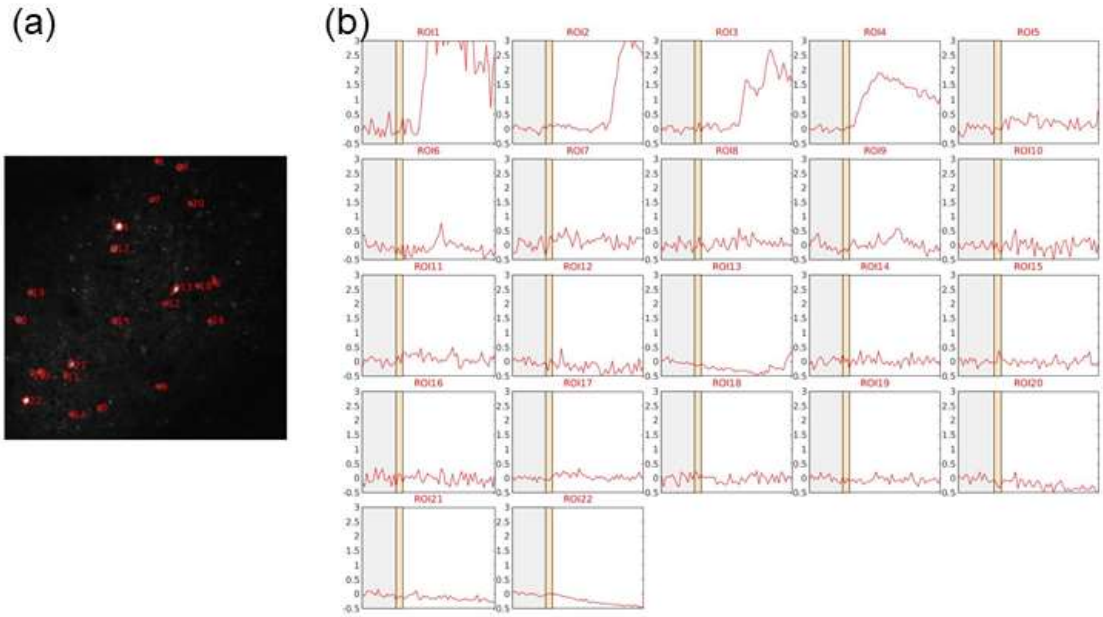


Figure 4: Offline identification of one field of view by HNCcorr. (a) summary image of one example FOV. (b) local correlation map of the same FOV, which HNCcorr was based on. (c) segmented mask by HNCcorr. Red, summary image; Green, segmented components with low baseline activity; Yellow, segmented components with high baseline activity. (d) one example patch containing a cell, and its distinct correlation maps of pixels in or out of the cluster. (e) activities of three example pixels in the patch.

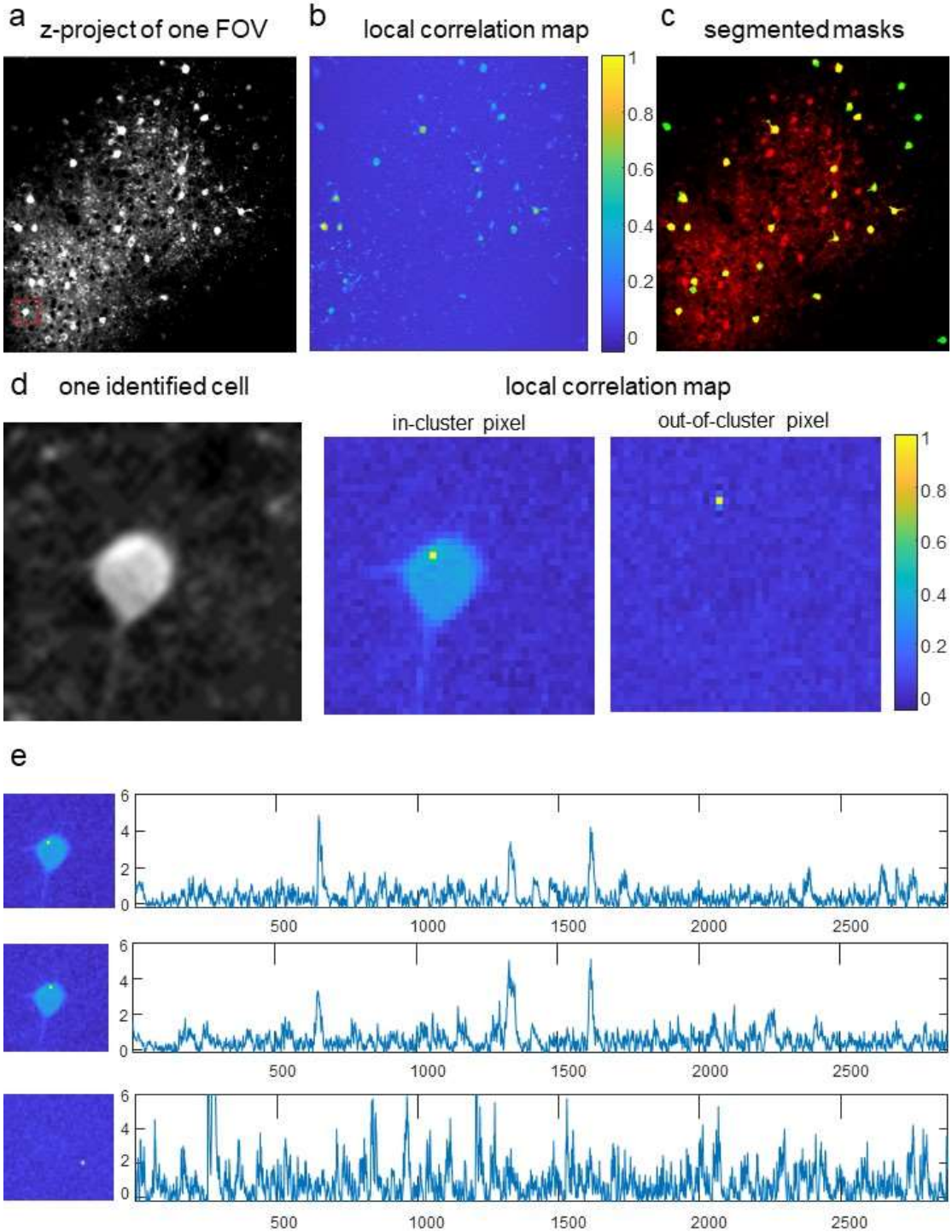


Figure 5: Cross session alignment using affine transformation. (a) left, two example sessions taken in different days; coloured left, overlay of two sessions; coloured right, overlay of same sessions aligned properly. In coloured images: red, session 1; cyan, session 2. (b) zoomed-in view of two areas taken from overlays in (a). Left, unaligned; Right, aligned. (c) all sessions aligned and summarised together. (d) all segmented masks merged into one, and its mapping on the summary image. (e-f) two example neurons that exhibits differently across sessions.

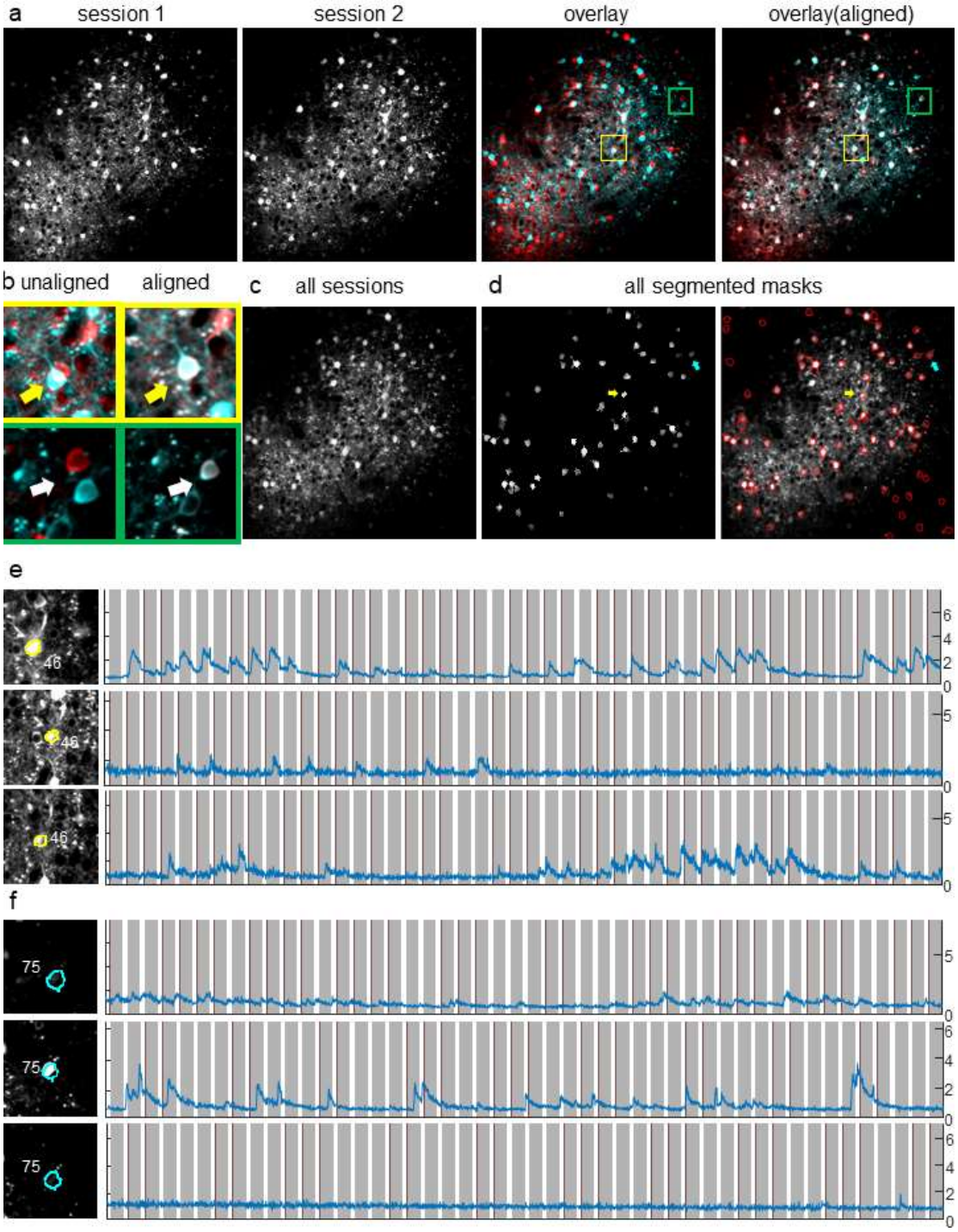
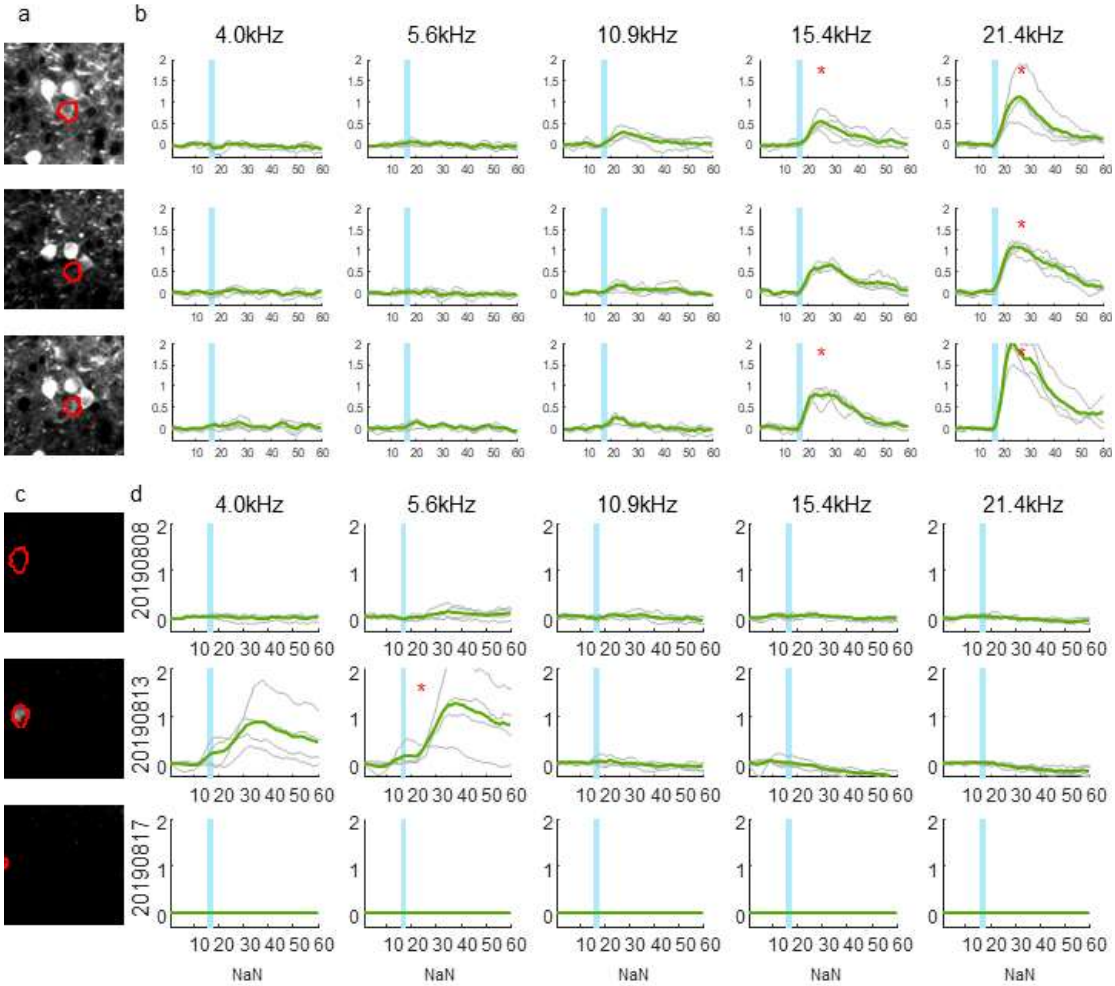


Figure 6. Activities of two example ROIs across all sessions. (a) one example neuron existing in the central part of the image, and (b) its activity across three sessions. Red asterisk indicates a three-fold change in the standard deviation after and before the stimulus. (c-d) another example neuron that partially existed only in one session.



Box 1. The algorithm of our registration.

```
function [shifts, registered] = register_stack(moving, template, maxshift)

    imagesize = size(moving);
    template = normalise(template); % normalised by mean-centralisation

    tflip = rotate(template, 180 degrees);
    tfliplarge = expandzeros(tflip, maxshift in 4 directions);
    sumtpl2 = 2Dconvolution(tfliplarge^2, ones(imagesize), 'valid');

    for a = each frame in moving

        a = normalise(a);
        alarge = expandzeros(a, maxshift in 4 directions);
        suma2 = 2Dconvolution(alarge, ones(imagesize), 'valid');
        sumat = 2Dconvolution(alarge, tflip, 'valid');

        val = suma2 + sumt2 - 2*sumat;
        [min_diff, shifts] = min(val);

        shifts = upsample_local_search(shifts);
        registered(this frame) = move_rigid(a, -shifts);
    end
end

function y = conv2_fftvalid(x, mshift, fsize)
% A simplified implementation of conv2.
    y = ifft2(fft2(x) .* fft2(mshift));
    y = real(y(1:fsize(1), 1:fsize(2)));  % trim to proper size
end

function mout = cshift(m, px)
% circularly shift the matrix in px pixels so that it matches conv2_fftvalid
    mout = zeros(size(m)+px);
    mout = circular_shift(m, -px in both direction);
end
```

Box 2. Online segmentation of components showing significant activity.

```
function map = Online_Segmentation(block_of_video, thresholds)
    % segment part
    block = centralised block_of_video;
    sum_activity = sum(block , 3);
    sum_activity(sum_activity&lt;0) = 0;
    scaling = std(block, [], 3);
    map = mat2gray(sum_activity .* scaling);

    % filter part
    map = im2bw(map, depth_threshold to keep pixels);
     map(areas smaller than area_threshold) = 0;

    convex_map = convexhull(map);  % use convex hull to make meaningful areas
    map = labelmatrix(convex_map);
end
```

Box 3. ORACA online processing.

```
function online_processing(trial_data, trial_info)
    % registration:
    trial_data = register_stack(trial_data, template=mean(trial_data,3));
    % identification:
    roimap = glance_std(trial_data);
    [labelmap, stats] = glance_filter(wmap);
    roimap = glance_merge_convex(stats);
    % extraction:
    trial_traces = trace_extract_trial(trial_data, roimap, trial_info);
    % display:
    uionline_show_trial(images={mean(trial_data,3),roimap}, trial_traces);
end
```

Box 4. Cross session alignment algorithms.

```
function cross_session_alignment(files)

    trials = add all files('mouseID', 'FOVn', 'dates', {}, 'files', {});

    for currentTrial = 1:length(trials)
        [mouseID, FOVn, file] = trials(currentTrial);
        first_session = load earliest file in trials(currentTrial);
        summary_image(1) = summary(first_session);

        if more sessions available
            for dates = 2:length(trials(currentTrial))
                later_session = load next session of same [mouseID, FOVn];
                t_concord = ProjectionCorrection(first_session, later_session);
                if ~isempty(t_concord)
                    % effective alignment
                    summary_later = summary(later_session);
                    summary_image(end+1) = transform(summary_later, t_concord);
                end
            end
            summary_image = stack mean(summary_image);
            save_image(summary_image, [mouseID '_' FOVn '.projection'])
        end
    end
end

function t_concord = ProjectionCorrection(first_session, later_session)

    summary_first = summary(first_session);
    summary_later = summary(later_session);

    % automatically align two images using rigid first
        t_concord = imregtform(later_session, first_session, 'rigid', ...);
        summary_later_trfrmd = transform(summary_later, t_concord);
        confirmed = InteractiveConfirm(summary_first, summary_later_trfrmd);
    if confirmed, return; end

    % try affine again if rigid do not meet our demand
        t_concord = imregtform(later_session, first_session, 'affine', ...);
        summary_later_trfrmd = transform(summary_later, t_concord);
        confirmed = InteractiveConfirm(summary_first, summary_later_trfrmd);
    if confirmed, return; end

    % do this manually
    while ~confirmed
        [newCoords, oldCoords] = cpselect(later_session, first_session);

        adjust newCoords by cpcorr;
```

```
        t_concord = fitgeotrans(newCoords,oldCoords,'projective');
        summary_later_trfrmd = transform(summary_later, t_concord);
        confirmed = InteractiveConfirm(summary_first, summary_later_trfrmd);
    end
end
```