S1. Datasets

S3D dataset The manually labeled S3D dataset is conceived to train or evaluate spine detection and depth-tracking models. In Table S1 we show an overview of its content, reporting the overall number of detections in the 2D images (slices), the overall number of 2D images, 3D volumes, and 3D spine instances (tracks), as well as per volume. We show this for the entire data set (total) and for one representative, randomly generated train/validation/test partitioning. The dataset is divided into training, validation, and test sets, based on the number of detections, allocating at least 80% to training, at least 9% to validation, and the remainder to testing. To prevent data leakage, we ensure that observations from the same field of view (regardless of timepoint) were never split across different partitions.

	Training	Validation	Test	Total
Detections	10585	1136	883	12604
Image slices	2676	551	331	3558
Volumes	121	22	16	159
Tracks	2060	198	195	2453
Slices per vol.	22 ± 8	25 ± 7	21 ± 2	22 ± 7
Detections per vol.	87 ± 89	52 ± 62	55 ± 48	79 ± 83
Tracks per vol.	17 ± 16	9 ± 10	12 ± 11	15 ± 15

Table S1. **Content information for dataset S3D**. We report the overall number of 2D detections, image slices, volumes, and tracks (unique 3D spine objects), and the mean ± standard deviations for the number of slices per volume, the number of detections per volume, and the number of tracks per volume. We show these numbers for the complete dataset (total) and one randomly generated partition into training, validation, and test sets.

Additional datasets for spine detection evaluation To evaluate our detection method, we use our S3D dataset alongside two open-source datasets (see Table S2). These external datasets vary in animal, brain region, sample condition (*in vivo*, *in vitro*, *ex vivo*) and image resolution, allowing us to assess the robustness and generalizability of our approach across diverse experimental conditions.

S2D+T dataset The annotations of the S2D+T dataset are produced with the goal to develop and to validate spine time-tracking methods. An overview of its content is shown in Table S3, reporting the number of total 2D images (slices), 3D volumes, 2D detections, and 3D spine objects (tracks) in total, per volume, and per image.

S2. Implementation details

S2.1. Dendritic spine detection in 2D

Deformable DETR training Our Deformable DETR detection model is trained for a total of 80 epochs, and the selected checkpoint is the one with the highest score for the validation partition. The detailed data partitioning is shown in Table S1. We use image flipping as data augmentation techniques, and stochastic gradient descent as optimizer, $1e{-}3$ as the learning rate, and $3e{-}6$ as the weight decay. Our Deformable DETR model is implemented in PyTorch and extends the work of Vogel et al. [5]. Our model is trained on one NVIDIA GPU A100.

Dataset	Imaging technique	Brain region	Resolution (µm per px)	Sample condition
DeepD3 Dataset [3]	Two-photon	Rat hippocampus CA1	$0.094 \times 0.094 \times 0.5$	*Ex vivo*
Smirnov-Garrett [1]	Two-photon	Mouse hippocampus CA1	0.067×0.067	*In vitro*
S3D	Two-photon	Mouse auditory cortex	$0.107 \times 0.107 \times 0.5$	*In vivo*

Table S2. **Overview of the three datasets used for the evaluation of automated spine detection.** Besides our data (S3D), the DeepD3 dataset and the Smirnov-Garrett dataset, two publicly available datasets are used to assess the generalizability of our spine detection method in comparison to previously published methods.

	Volumes	Image slices	Detections	Tracks
Total	720	7648	3974	1450
Per volume	-	11 ± 5	6 ± 3	6 ± 3
Per slice	-	-	1 ± 0.45	1 ± 0.45

Table S3. **Content information for dataset S2D+T.** Number of volumes, image slices, detections and tracks for the overall dataset (total), per volume at a single timepoint, and per slice. Detections refer to the 2D polygons, and tracks refer to the number of unique 3D objects (each represented with a single 2D instance) that were manually tracked. Note that here the number of detections in a volume (at a single timepoint) equals that of tracks, since for each 3D object only one 2D instance was labeled.

Prediction of spine bounding boxes on 2D When predicting with our detection model, the parameters used are kept fixed, and our minimum confidence threshold set to 0.5. We use DeepD3's 32F model to predict binary masks of spines and built ROIs using their GUI with the default parameter values, except for the image resolution, which we adjust as specified by the dataset. In the case of the Vogel et al. model, we adjust only the minimum confidence threshold, setting it to 0.5 (as per default) for the S3D-test dataset, and to 0.1 for the DeepD3 and Smirnov datasets, as the images are much dimmer compared to the data it was trained on.

Data pre-processing for evaluation To enable evaluation when using the segmentation-based model DeepD3 [2], we first convert the binary masks into individual 2D object instances by extracting the minimum bound box that enclos-

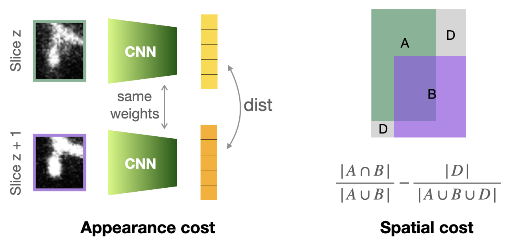

Figure S1. **Schematic of the cost terms to resolve matching across depth. Left**) Our Siamese network trained on spine data is used to encode a 2D image patch of a spine into an feature embedding. Pairs of 2D instances that potentially belong to the same 3D object are encoded and the Euclidean distance between their embeddings is computed. **Right**) To measure spatial consistency, the generalized intersection over union between the bounding boxes (here represented as A and B) of the 2D spines is calculated.

ing each mask. In the case of SpineS, each predicted SURF feature corresponds to a single 2D point. Thus, we assign each prediction a box of fixed size, based on the average dimensions of ground-truth boxes from the evaluation dataset.

S2.2. Spine tracking across depth

Siamese network training As mentioned in Section 2.4, a CNN is trained to encode visual features representative of the same 3D spine instance. From the annotations produced for the train partition of the S3D dataset, 2D image patches of spines are extracted using their bounding box coordinates. Each image patch is labeled with its corresponding 3D identity across depth.

The training is done for a maximum of 100 epochs, with 20 epochs of patience for early stopping. Gaussian blur and color jittering are used as augmentation techniques, dropout was used to avoid overfitting, and Adam as the optimizer. The learning rate is set to $1\mathrm{e}{-4}$, and the weight decay to $1\mathrm{e}{-5}$.

Trackformer training Trackformer is trained across 80 epochs, starting from the pre-trained version for the MOT17 dataset, with Deformable DETR with iterative refinement as its detection module.

S2.3. Automated computation of spine features

Spine size Given a 2D spine detection, an extended bounding box is created by taking 25 more pixels on each side, and the mean of the 20% darkest pixels is used to estimate a background threshold. Analogously, the mean of the 98% brightest pixels is calculated to estimate the dendrite intensity. All of the pixels below the background threshold in the original bounding box are set to 0. Then, a Gaussian weighting mask is applied to reduce the influence of possible bounding box overestimation. Next, the background threshold value is subtracted from the foreground pixels, and the pixels intensities inside the original bounding box are summed. Finally the integrated fluorescence is normalized by the dendrite intensity. After depth-tracking, we set the size of a 3D spine object as the median across the sizes of its 2D instances.

Spine-to-dendrite distance The spine-to-dendrite distance is estimated through a two stage process. First, the spine subregion (delimited by the bounding box) is binarized using Otsu thresholding and overlaid with a dilated binary mask of the non-spine surrounding region. The coordinate at the center of the overlapping area is then taken as the spine-dendrite junction [4]. Second, the largest contour of the binarized spine region is extracted, and an ellipse is fitted to it. The center of this ellipse is used as the spine head center. The distance between these to points defined our estimate of the spine-to-dendrite distance.

References

[1] Labeled Dendritic Spines - Training Data, 2018. 1

[2] Martin HP Fernholz, Drago A Guggiana Nilo, Tobias Bonhoeffer, and Andreas M Kist. Deepd3, an open framework for automated quantification of dendritic spines. *PLOS Computational Biology*, 20(2):e1011774, 2024. 1

[3] Martin H P Fernholz, Drago A Guggiana Nilo, Tobias Bonhoeffer, and Andreas M Kist. DeepD3 Datasets, 2023. 1

[4] Yutaro Kashiwagi, Takahito Higashi, Kazuki Obashi, Yuka Sato, Noboru H Komiyama, Seth GN Grant, and Shigeo Okabe. Computational geometry analysis of dendritic spines by structured illumination microscopy. *Nature communications*, 10(1):1285, 2019. 2

[5] Fabian W Vogel, Sercan Alipek, Jens-Bastian Eppler, Pamela Osuna-Vargas, Jochen Triesch, Diane Bissen, Amparo Acker-Palmer, Simon Rumpel, and Matthias Kaschube. Utilizing 2d-region-based cnns for automatic dendritic spine detection in 3d live cell imaging. *Scientific reports*, 13(1):20497, 2023. 1