




A new training strategy for spatial transform networks (STN's)

Francisco Navarro Moya¹ · Joan Carles Puchalt¹ · Pablo E. Layana Castro¹ · Antonio García Garví¹ · Antonio-José Sánchez-Salmerón¹ 

Received: 1 October 2021 / Accepted: 30 January 2022 / Published online: 1 March 2022
© The Author(s) 2022

Abstract

Spatial transform networks (STN) are widely used since they can transform images captured from different viewpoints to obtain an objective image. These networks use an image captured from any viewpoint as input and the desired image as a label. Usually, these images are segmented, but this could lead to convergence problems if the percentage of overlap between the segmented images is quite low. In this paper, we propose a new training method to facilitate the convergence of a STN in these cases, even when there is no overlap between the object's projections in the two images. This new strategy is based on the incorporation of the distance transformation images to the training, thus increasing the useful image information to provide gradients in the loss function. This new training strategy has been applied to a real case, with images of *Caenorhabditis elegans*, and to a simulated case, which uses artificial images to ensure that there is no overlap between the images used for the assays. In the assays carried out with these datasets, we have shown that the training convergence is strengthened, reaching a precision level for IoU metric of 0.862 and 0.984, respectively, and the computational cost has been maintained compared to the assay with segmented images, for the real case.

Keywords Spatial transform network · Learning strategy · *Caenorhabditis elegans* · Correspondence application

1 Introduction

Since Jaderberg et al. [1] developed the spatial transformer network (STN), it has been employed to solve a multitude of tasks. In most cases, it is used as an intermediate phase that adapts the input images to tackle problems with greater efficiency. Among some of the most frequent problems in which this type of network is used, are the classification task [1–4], the object detection task [5–8] or the correspondence task [9, 10].

On developing STNs, Jaderberg et al. [1] presented its use for the classification task, in which the objective of the STN is to transform the input image to make the study element more easily identifiable, in this case a dataset such as MNIST or Street View House Numbers (SVHN) to identify numbers and so on CUB-200-2011 birds dataset to identify the parts of birds. As in the two first datasets, the

STN were also used in [2] to classify numbers printed on football players' jerseys. And, beyond the classification of numbers the STNs have been used in many other areas such as the classification of traffic signs in [3], or, in the field of the medicine, to recognize tumor cells in [4].

For the object detection task, the purpose of STN is to transform the image so that it is easier to recognize the characteristics of the object to be identified. For example, in [5] thermal images of power stations were transformed to detect the different elements, or in [6] where the people in images captured with fish-eyes cameras were detected. They have also been used to detect a section of images that have some special characteristic in [7]. Or for crowd counting in [8], but in this case the STN have been applied to a video dataset instead of an image dataset. To do so, they divide the video frames into blocks and seek to predict the trajectory of the people in each block. Then they apply the corresponding transformation and compare this new activation map with the ground truth for the next frame, if it differs, it means that the crowd has changed.

Finally, regarding the correspondence task, which is the application studied in this paper, the STNs try to modify the appearance of an image to transform into another

✉ Antonio-José Sánchez-Salmerón
asanchez@isa.upv.es

¹ Universitat Politècnica de València, Camino de Vera S/N,
46022 Valencia, Spain

appearance used as a ground truth. This application is explained in [9], which seeks to give a more realistic appearance to synthetic images of the fruit flies *D. rerio* and the worm *C. elegans*. To do so, synthetic images have been segmented as input and real images segmented as target, thus, the STN calculates the transformation required to give to the synthetic images the appearance of real images. And we also find this problematic in [10], where researches try to obtain a 3D model of a face from information taken from images of the same person from different viewpoints.

Here, the problem we study is similar to that posed in [9], in that the STN is used to transform images by segmenting the object of interest. But unlike [9], here we seek to transform images in which the overlap of the object's projections between the input images and the target image is not ensured.

Images of *C. elegans* have been used as a dataset. These were obtained using two micro-cameras with different viewpoints. Of these two cameras, one of them, which will be referred to as Micro 1 from hereon, captures images in which the worm is centered, while the second, which we will call Micro 2, captures images in which the worm has been displaced slightly toward the top right corner.

The principal objective of the STN is to give the images obtained by Micro 2 a similar appearance to those of Micro 1, to solve the correspondence problem and to be able to use these images in future works.

In our case, as seen in Fig. 1, the worm represents a small portion of the images there may be no overlap between the projections of the two worms depending on their position.

To avoid this, in the first instance, it was tried to use images of distances to the edges of the object to perform the training. The use of this type of images, which is novel in itself, increase the useful image information allowing convergence in these cases, but the precision is reduced.

For this reason, several new strategies have been proposed to favor the convergence in these cases and avoid this loss of precision. The one that gives the best result consists of a hybrid strategy that begins training alternately with transformed distance images and segmented images and switches to only segmented images to achieve better precision when convergence is already assured.

2 Methods

2.1 Network architecture

The structure of the STN is divided in two different parts. On the one hand, a localization neural network, and on the other hand, a grid generator that uses the result of the first one to obtain the grid and apply it to the corresponding images to obtain the desired image, as can be seen in Fig. 2.

The network employed for this study was the same as in [9]. In it, the first part is responsible for locating the pixels of the object of interest within the image. In turn, the localization network can be divided into two stages.

The first stage is made up of a sequence of convolutional layers, each followed by a maxpooling layer and by a SeLU activation layer, except the last one, which is only followed by a SeLU activation layer. Therefore, this part has a total of five convolutional layers, five SeLU activation layers and four maxpooling layers, in the order shown. Table 1 shows a summary of the main characteristics of each layer.

Then its second section adapts the information obtained in the previous stage to define the parameters of the transformation to be performed. This section is made up by two fully connected layers, between which a SeLU activation layer is placed. The result of the last layer is a vector of 12 elements, whose values are reorganized in a matrix of 3 rows and 4 columns which represent the transformation

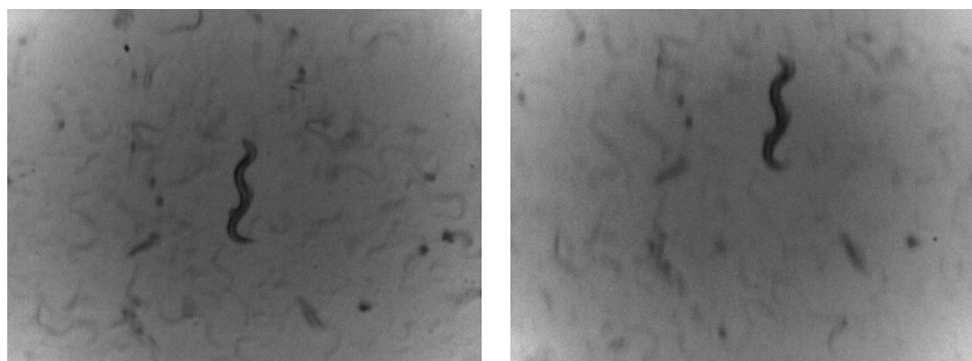


Fig. 1 Images obtained from Micro 1 camera (left) and Micro 2 camera (right). These images are not exactly the images captured by the cameras, but have had a filter applied to increase the contrast

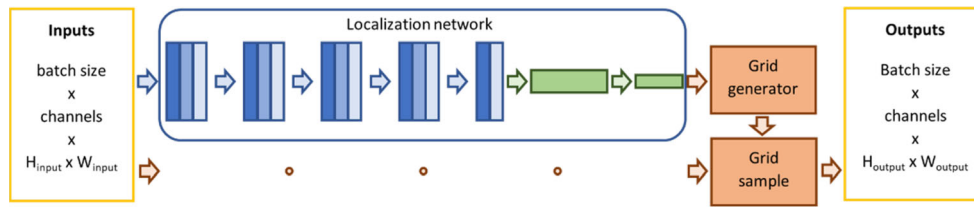


Fig. 2 Scheme of a spatial transform network (STN)

Table 1 Summary of layers features

Layer	Output size	Layer details
Conv 1	[Batch size, 8, 1938, 2586]	Kernel = 7 × 7
MaxPooling 1	[Batch size, 8, 969, 1293]	Stride = 2
Activation SELU 1	[Batch size, 8, 969, 1293]	Inplace = True
Conv 2	[Batch size, 10, 965, 1289]	Kernel = 5 × 5
MaxPooling 2	[Batch size, 10, 482, 644]	Stride = 2
Activation SELU 2	[Batch size, 10, 482, 644]	Inplace = True
Conv 3	[Batch size, 20, 478, 640]	Kernel = 3 × 3
MaxPooling 3	[Batch size, 20, 239, 320]	Stride = 2
Activation SELU 3	[Batch size, 20, 239, 320]	Inplace = True
Conv 4	[Batch size, 40, 237, 318]	Kernel = 3 × 3
MaxPooling 4	[batch size, 40, 118, 159]	Stride = 2
Activation SELU 4	[Batch size, 40, 118, 159]	Inplace = True
Conv 5	[Batch size, 10, 116, 157]	Kernel = 3 × 3
Activation SELU 5	[Batch size, 10, 116, 157]	Inplace = True
Linear 1	[Batch size, 32]	–
Activation SELU 6	[Batch size, 32]	Inplace = True
Linear 2	[Batch size, 12]	–

to be carried out (Eq. 1). This matrix serves as input for the grid generator.

Finally, the second section of the STN uses the information provided by the localization neural network to obtain the grid to be applied on the input images in order to make it resemble the target image. In this case, two PyTorch functions have been used: “affine_grid” and “grid_sample”.

The former, “affine_grid”, is in charge of obtaining the grid. As inputs this function uses the 3 × 4 matrix obtained in the localization network and the grid size to be obtained, which must be the same as the target images size.

The latter, “grid_sample”, obtains the final transformed image, using the grid to be applied and the input image, through a bilinear interpolation.

These two functions allow the calculation of two types of transformations, on the one hand, affine transformations and, on the other hand, projective transformations. In our case, projective transformation has been used, since this is the most appropriate transformation for the images employed, which is why the “affine_grid” input has the

size 3 × 4, in the case that the affine transformation is used, this matrix must have size 2 × 3.

$$\begin{pmatrix} X_{input} \\ Y_{input} \\ Z_{input} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix} \cdot \begin{pmatrix} X_{output} \\ Y_{output} \\ Z_{output} \end{pmatrix} \quad (1)$$

Also, in the “grid_sample” function, the “padding_mode” parameter has been set to “edge” to extend the colors of the edge pixels to the pixels that are outside the image instead of turning them black.

2.2 Simulator

A virtual camera simulator has been developed to generate synthetic images, in order to control the characteristics of the network input images. It should be noted that the main function of the simulator is not to expand the real images dataset, but to generate an alternative dataset to analyze specific cases.

The simulator enables the projection of the points defining the object of interest by means of the projection matrix extracted from the camera features (focal length, height, inclination...), so that in the resulting image it projects the object of interest according to the camera features.

First the morphology of the object to be captured must be defined, for which the points of its silhouette must be defined. As the shape of the object is not a critical aspect, to simplify its definition, a rectangle was generated, since it could be defined by the four vertices. In addition to its shape, its size must also be defined, which is set at a number of pixels similar to that of *C. elegans* in real images, i.e., approximately 80 × 300 pixels.

Subsequently, in order to grant greater variability to the images, certain parameters were established to randomly change the characteristics of the rectangles. On the one hand, their size can vary from 50% to 150% of the base size in each of the directions independently. On the other hand, the rectangle that is originally oriented vertically is randomly rotated around its center. And finally, as in the real images the positioning of the worm in Micro 1 is not fully centred, a small random displacement has also been added in each of the two axes.

Once the object represented by the simulator has been defined, the next step is to determine the projection matrices of the two cameras. To do this, the cameras were calibrated; ensuring that the image produced has a similar appearance to the real images.

Consequently, the simulator is capable of generating images of rectangles with a random position and shape captured from the perspective of the two cameras. A sample of these images is shown in Fig. 3.

2.3 Dataset

From the viewpoint of the dataset, there are two types, on the one hand, the dataset of real images, and on the other, the dataset of the simulated images, as already mentioned in the previous section, both cases will be analyzed separately.

2.3.1 Dataset of *C. elegans*

The *C. elegans* dataset was obtained by image capture system [11] in the laboratory. To capture the images two cameras were used (Micro 1 and Micro 2). These cameras take captures with a resolution of 1944×2592 pixels.

The images used as network inputs (Micro 2) maintain their original capture size, but in order to reduce memory use and ensure that all the information present in the target image is found in the initial image, Micro 1 images have been cropped to half their size, thus target image size is 972×1296 pixels.

Given the costly capture process, the *C. elegans* dataset had a total of 975 pairs of images. The 975 image pairs with a similar appearance to those shown in Fig. 1 were processed to obtain the segmented images and distance transformation using the functionalities of the OpenCV library.

Depending on the assays, the dataset may be formed by pairs of segmented *C. elegans* images (Fig. 4a, b) or by

pairs of distance-to-the-edge transformed *C. elegans* images (Fig. 4c, d), or by both types, depending on the case.

2.3.2 Synthetic dataset

The synthetic dataset has 1000 pairs of images, approximately the same as the *C. elegans* dataset. And, as in the case of the *C. elegans* dataset there are two types of images, segmented images and distance-transformed images.

By default, the simulator returns the segmented images of the rectangles. To obtain the distance-transformed images, a processing stage was required. A sample of the images returned by the simulator are those represented in Fig. 5a–d. These images have been grouped identically to the *C. elegans* dataset depending on the case studied.

2.4 Metrics

Different types of metrics were selected to estimate the performance of the assays; all of them have been applied on segmented images since the main objective is to transform the Micro 2 image so that the segmentations of the worms or the rectangles coincide.

On the one hand, because of the size of the object of interest, the number of non-coincident pixels between the two images was counted (Eq. 2) and the mean, maximum, and minimum values of all pairs of images were recorded. In this way, the error was better characterized.

$$\text{error} = \text{sum}(|X - Y|) \quad (2)$$

Where “X” and “Y” are the resulting segmented image and the target segmented image, respectively.

On the other hand, we decided to use the IoU (Intersection over Union) metric, since it reflects exactly what we set out to achieve. This metric calculates the ratio between the intersection and the union of the elements of the two segmented images, as reflected in Eq. 3.

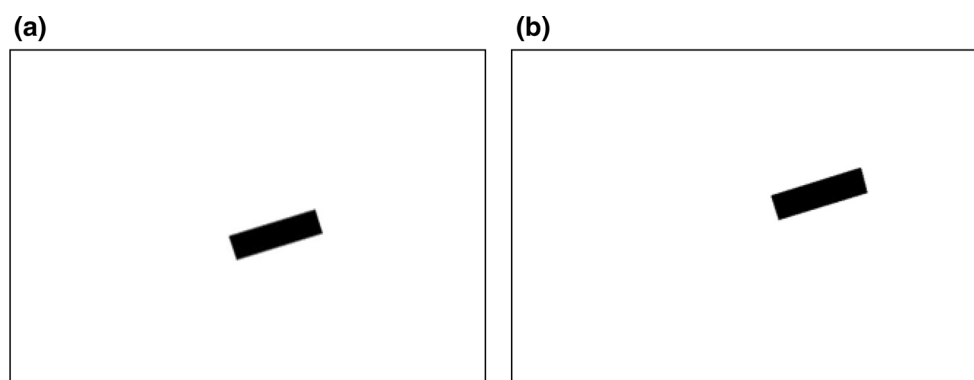


Fig. 3 Images obtained by the simulator. **a** Image Micro 1. **b** Image Micro 2

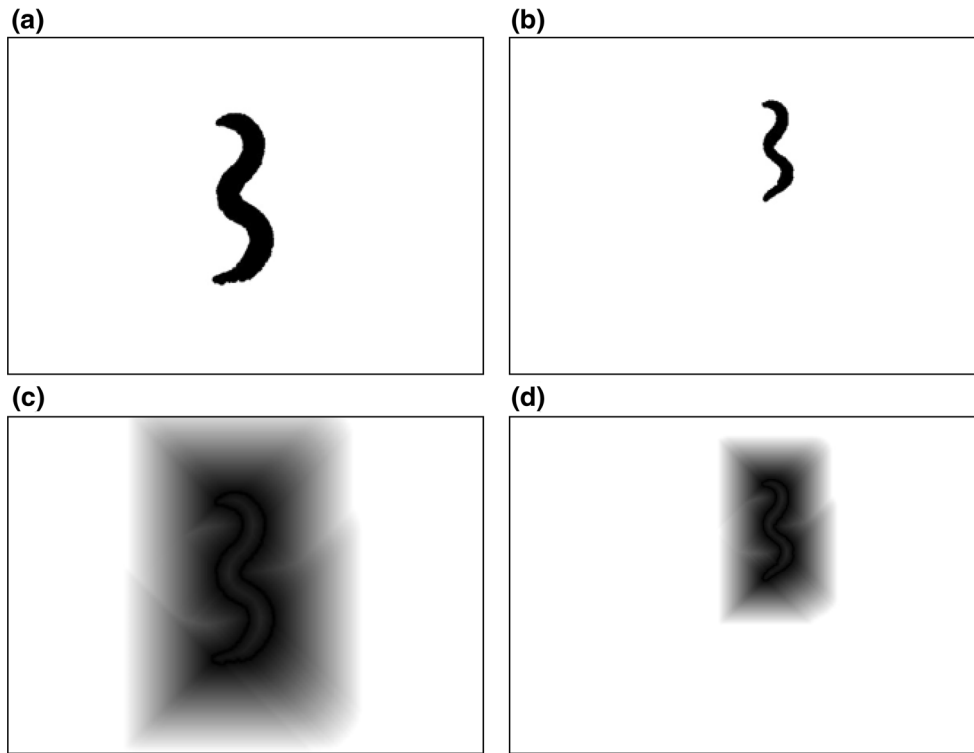


Fig. 4 Images of *C. elegans* dataset. **A** cropped segmented image Micro 1. **B** segmented image Micro 2. **C** cropped distance-transformed image Micro 1. **D** distance-transformed image Micro 2

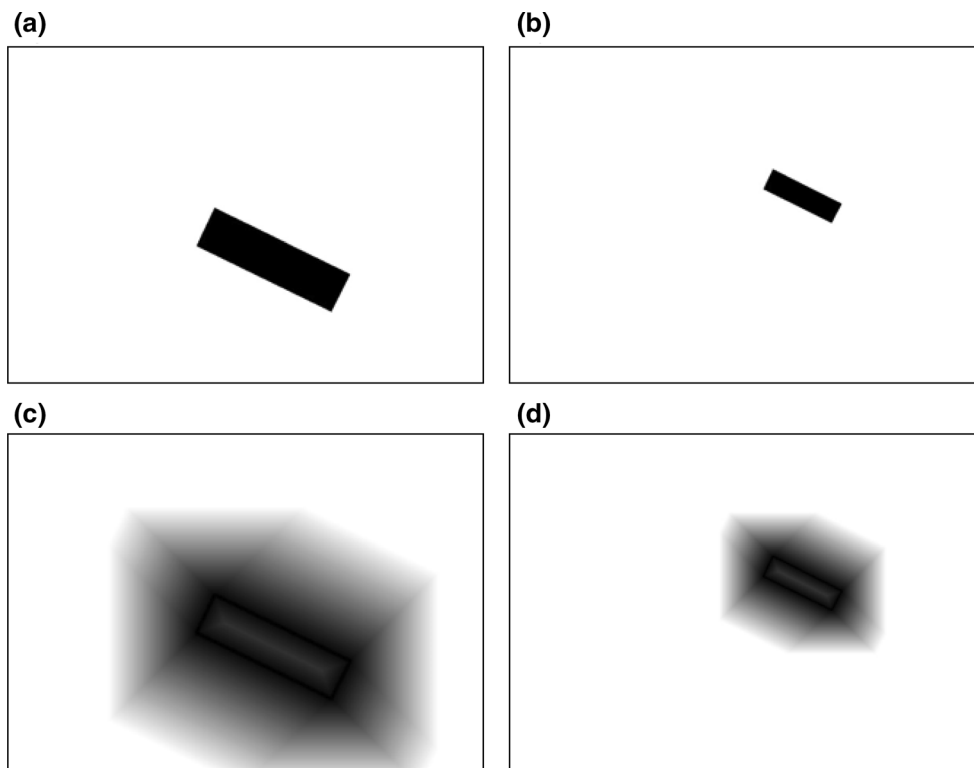


Fig. 5 Images of synthetic dataset. **A** cropped segmented image Micro 1. **B** segmented image Micro 2. **C** cropped distance-transformed image Micro 1. **D** distance-transformed image Micro 2

$$\text{IoU} = \frac{X \cap Y}{X \cup Y} \quad (3)$$

where “ X ” is the segmented worm of the transformed image and “ Y ” is the segmented worm of the target image. This ratio provides the percentage of overlap that occurs between the worms in the two images.

Lastly, in the assays with *C. elegans* images the training time has also been monitored to consider the computational cost, since if this is too high, a good alternative in terms of precision could be unfeasible due to such cost.

2.5 Hyper-parameter tuning

Hyper-parameter tuning is a fundamental task to ensure the convergence of a network. In our case, we look at the limitations of the network and the GPU to define them.

The first hyper-parameter to be fixed was batch size, which was set at eight due to the large size of the images, since this was the maximum size that met the memory limitation of the GPU used.

Regarding the optimizer, an Adam optimizer (Eq. 4) was chosen to perform the convergence, whose learning rate value had been defined with one of the assays carried out.

$$\begin{aligned} m &= \beta_1 \cdot m + (1 - \beta_1) \cdot \nabla_{\theta}(\mathcal{L}(\theta_t)) \\ v &= \beta_2 \cdot v + (1 - \beta_2) \cdot \nabla_{\theta}(\mathcal{L}(\theta_t))^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta \cdot m}{\sqrt{v \cdot \varepsilon}} \end{aligned} \quad (4)$$

where “ β_1 ” and “ β_2 ” are the attenuation coefficients of the moments of the first and second order respectively (the values of 0.9 and 0.99 are used, respectively), “ ε ” is the step value (which is usually a very small value), “ θ ” are the parameters of the network, “ t ” indicates the iteration in which we are, “ η ” is the learning rate and “ \mathcal{L} ” represents the loss function.

To define the stopping criterion, two different methods were followed depending on the dataset used. If the assay used the synthetic dataset, the training stopped after 150 times, while if the assay worked with the *C. elegans* dataset, it stopped when a given level of convergence was

reached, which corresponded to a value for the IoU of 0.86 (this value is very close to the upper limit of precision), or when the training was unable to improve further, this occurs when it fails to improve in the last 20 times.

And the last one to be fixed was the loss function. To train the STN, the loss function used was the Mean Square Error (MSE) loss, with the formula represented in 5.

$$\text{MSE}_{\text{loss}} = \frac{\sum_{i=0}^n \sum_{j=0}^m (X_{(i,j,c)} - Y_{(i,j,c)})^2}{m \cdot n} \quad (5)$$

where “ n ” and “ m ” are the size of the images for each of their axes, “ c ” is the channel of the images and “ $X(n, m, c)$ ” and “ $Y(n, m, c)$ ” the values of the pixels of the transformed image and the target image respectively. This function calculates the mean of the distances between the values of all the pixels of the two images, thus evaluating how similar they are.

3 Training methods, experiments and results

This section presents the methods developed together with the assays that have been performed and the results obtained. First, the assays in which only one type of image (segmented or distance-transformed) is used have been compared to determine which generates better results, and then some alternatives have been considered to improve the training performance.

3.1 Segmented image method compared to distance-transformed image method

3.1.1 Synthetic dataset

These two cases have been studied using the synthetic dataset to analyze some specific cases. For these assays, the computational cost was not calculated, as the aim was to find the method that provides the best performance, in terms of precision and robustness. In Algorithm 1, is shown the procedure of a training step for these assays.

Algorithm 1: Procedure of a training step

- 1: **Input** weights ϑ , optimizer opt , batch size B , training set D , image img , target $target$, model $model$, loss function $loss_fc$
 - 2: **Fetch** a batch B from D
 - 3: **Get** $grid$ from $model(img)$
 - 4: **Get** out from $grid_sample(grid, img)$
 - 5: **Get** $loss$ from $loss_fc(out, target)$ by calculating (Eq. 5)
 - 6: **Update** ϑ from $opt(\nabla loss)$ by calculating (Eq. 4)
-

Table 2 Assays carried out with synthetic images

	Mean error (px)	Min. error (px)	Max. error (px)	IoU
Segmented $5e^{-5}$	869	148	2172	0.98494
Segmented $1e^{-4}$	559	72	1390	0.99040
Segmented $5e^{-4}$	–	–	–	–
Segmented	714	72	2172	0.98767
Distances $5e^{-5}$	2358	757	4441	0.96074
Distances $1e^{-4}$	2373	602	4584	0.96055
Distances $5e^{-4}$	2440	912	4812	0.95935
Distances	2381	602	4812	0.96039

Apart from comparing the type of image used (segmented or distance-transformed), these assays have also served to adjust the learning rate with which the optimizer will work, for this reason assays have been carried out varying this value for both strategies.

Specifically, to carry out the assays, three different values for this parameter have been chosen, $5e^{-5}$, $1e^{-4}$ y $5e^{-4}$. All these assays have been repeated three times to estimate the mean of the results. These results are shown in Table 2.

Table 2 clearly shows that working with segmented images obtains better precision, moreover precision is better than that obtained using any distances assay. This large difference may be mostly, due to the fact that the distance-transformed images do not fully represent real distance, but rather an approximation with interpolations.

Regarding the learning rate values, it is remarkable that with the highest, the network is unable to converge when working with segmented images, while it does achieve this when working with distance-transformed images.

For the remaining values, although there is not much differentiation as in the case of the types of images, it is observed that for the learning rate value of $1e^{-4}$ better results were obtained for the segmented-image assays. In the assays using the distance-transformed images, the change in learning rate value was practically indifferent. So, the learning rate value selected was $1e^{-4}$.

In addition to these assays, and to demonstrate which method provides greater robustness, an assay was

performed with rectangles that do not overlap in any of the image pairs from Micro 1 and Micro 2 cameras. To carry out this assay, the simulator configuration was maintained and only the size of the rectangles forming the images was reduced until clearly avoiding overlapping, as shown in Fig. 6.

As shown in Table 3, the fact there is no overlap, the assay carried out with the segmented image is unable to converge, while the distance-transformed images assay can converge as a result of the extra information provided by the distances to the edges of the objects of interest.

These examples show empirically that using distance-transformed images is more robust and can help to converge. By contrast if convergence is achieved, the segmented-image dataset provides greater accuracy.

3.1.2 Dataset of *C. elegans*

With the conclusions obtained in the synthetic images assays, the first assay that was carried out with *C. elegans* dataset was the verification of using segmented images the network can converge. As with the synthetic dataset (Tables 2, 3), these assays have been repeated to obtain more reliable results. These results can be seen in Table 4.

As shown in Table 4, this method can achieve convergence, reaching a precision of 0.863 for the IoU metric. Despite this, during the simulations it was also observed that in some cases the network was unable to converge, this may be due to the low initial overlap between the two images and to the usage of reduced batch size. In Fig. 7a, b

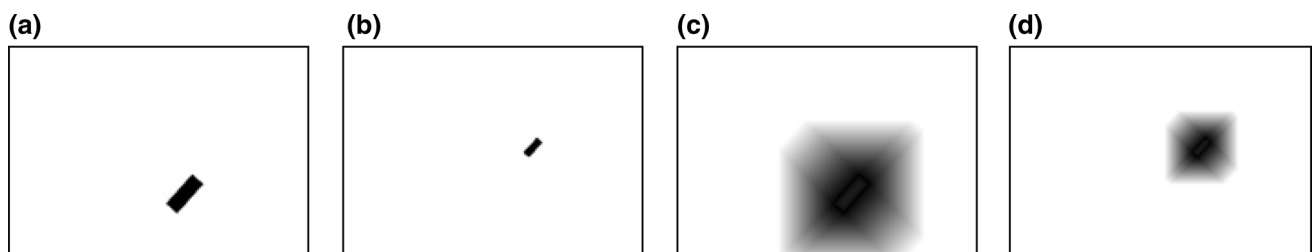


Fig. 6 Synthetic no overlap images dataset. **A** cropped segmented Micro 1 image. **B** Segmented Micro 2 image. **C** cropped distance-transformed Micro 1 image. **D** distance-transformed Micro 2 image

Table 3 Assay with no overlap synthetic images

	Mean error (px)	Min. error (px)	Max. error (px)	IoU
Segmented	–	–	–	–
Distances	1038	304	2276	0.91445

Table 4 Assays with *C. elegans* segmented images dataset

	Mean error (px)	Min. error (px)	Max. error (px)	IoU	Time (h)
Assay 1	4907	2020	29923	0.86299	3:44:30
Assay 2	4968	1978	19600	0.86130	4:01:30
Assay 3	4985	2270	29667	0.86184	3:58:00

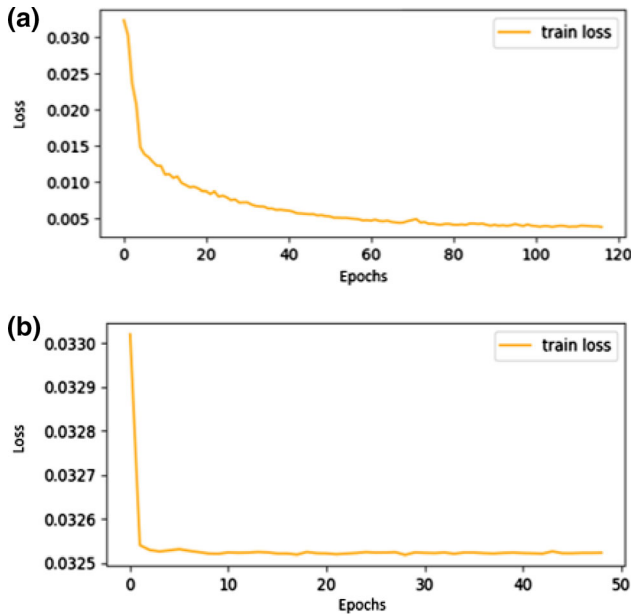


Fig. 7 Downward trends of assays with *C. elegans* segmented images. **a** Assay that has achieved the convergence. **b** Assay that has not achieved the convergence

shows the graphics of the loss function of an assay that achieved convergence and another that did not, respectively.

3.2 New training methods

For this reason, to take advantage of the robustness of the distance-transformed images without disregarding the precision of the segmented images, we decided to design a new mixed training strategy in which both types of images were used.

The first new strategy consists of using distance-transformed images in the first training times and, when a certain level of precision is reached, it is switched to the segmented images to achieve greater precision (Fig. 8), this training method is shown in Algorithm 2. Thus, we have attempted to avoid the convergence problems. The change of dataset was made when the loss function returned a value less than 0.017, which corresponds to an IoU value of around 0.6.

In the mixed method, apart from the change of the dataset, the optimizer was also changed, since, although in both cases the same loss function was used, the

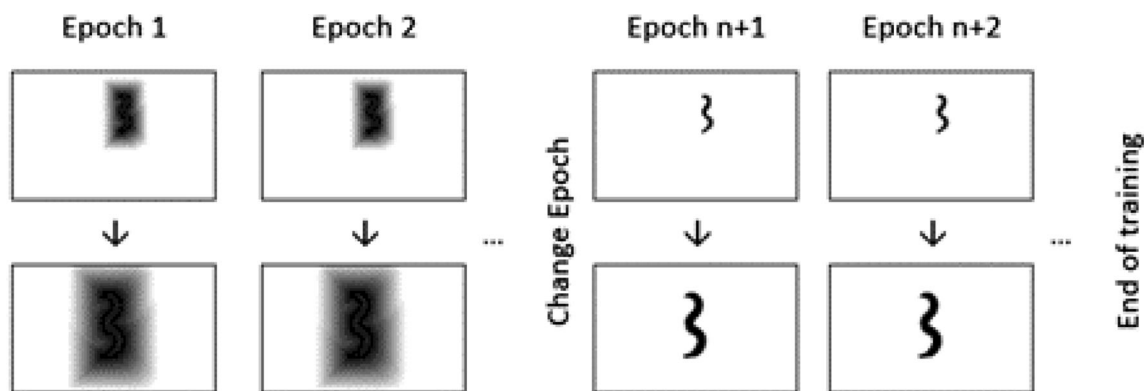


Fig. 8 Schema of mixed training method

Algorithm 2: Mixed training method

```

1: Input weights  $\vartheta$ , epochs  $E$ , batch size  $B$ , steps  $S$ , shift accuracy  $iou_c$ , maximum accuracy
    $iou_{max}$ , minimum loss  $loss_{min}$ , variable of training improvement  $not\_im$ 
2: Load segmented Micro 1 images  $target\_seg$ , segmented Micro 2 images  $seg$ , distance
   transform Micro 1 images  $target\_dist$ , distance transform Micro 2 images  $dist$ 
3: Initialize model  $model$ , optimizer  $opt1$ , loss function  $loss\_fc$ 
4: for  $e = 1, 2, 3, \dots, E$  do
5:   Shuffle training set  $D$ 
6:   If  $iou < iou_c$  then
7:     for  $s = 1, 2, 3, \dots, S$  do
8:       Update  $\vartheta$  by calling  $algorithm1(\vartheta, opt1, B, D, dist, target\_dist, model, loss\_fc)$ 
9:       Get  $out2$  from  $grid\_sample(grid, seg)$ 
10:      Get  $iou$  by calculating (3)
11:     end for
12:   else
13:     If first time inicialice optimizer  $opt2$ 
14:       for  $s = 1, 2, 3, \dots, S$  do
15:         Update  $\vartheta$  by calling  $algorithm1(\vartheta, opt1, B, D, dist, target\_dist, model, loss\_fc)$ 
16:         Get  $iou$  by calculating (3)
17:       end for
18:       If  $loss < loss_{min}$  then
19:         Update  $loss_{min}$ 
20:         Reset  $not\_im$ 
21:       else
22:         Update  $not\_im += 1$ 
23:       end if
24:       If  $iou > iou_{max}$  or  $not\_im = 20$  break
25:       end if  $grid$  from  $model(img)$ 
26:     end for  $out$  from  $grid\_sample(grid, img)$ 

```

optimization problem changed depending on the type of the image. So, we used two optimizers, both of type Adam and with a learning rate value of $1e^{-4}$.

After seeing the results of the first method, a second new strategy was proposed to design a modified mixed strategy to increase the convergence. As in the mixed strategy, there is a differentiation between the first periods and the rest. But, in this case, in these early periods the network does not train solely with distance-transformed images, but

rather periods with distance transformed and segmented images have been intercalated (Fig. 10), this new training method is shown in Algorithm 3.

3.3 Experiments and results

For each of these methods, a set of experiments has been carried out with the *C. elegans* dataset, as shown in Tables 5 and 6, respectively.

Algorithm 3: Modified mixed training method

```

1: Input weights  $\vartheta$ , epochs  $E$ , batch size  $B$ , steps  $S$ , shift accuracy  $iou_c$ , maximum accuracy  $iou_{max}$ , minimum loss  $loss_{min}$ , variable of training improvement  $not\_im$ 
2: Load segmented Micro 1 images  $target\_seg$ , segmented Micro 2 images  $seg$ , distance transform Micro 1 images  $target\_dist$ , distance transform Micro 2 images  $dist$ 
3: Initialize model  $model$ , optimizer  $opt1$ , loss function  $loss\_fc$ 
4: for  $e = 1, 2, 3, \dots, E$  do
5:   Shuffle training set  $D$ 
6:   if  $iou < iou_c$  then
7:     if  $e \% 2 = 0$  then
8:       for  $s = 1, 2, 3, \dots, S$  do
9:         Update  $\vartheta$  by calling algorithm1( $\vartheta, opt1, B, D, dist, target\_dist, model, loss\_fc$ )
10:        Get  $out2$  from  $grid\_sample(grid, seg)$ 
11:        Get  $iou$  by calculating (3)
12:      end for
13:    else
14:      if first time inicialice optimizer  $opt2$ 
15:        for  $s = 1, 2, 3, \dots, S$  do
16:          Update  $\vartheta$  by calling algorithm1( $\vartheta, opt1, B, D, dist, target\_dist, model, loss\_fc$ )
17:          Get  $iou$  by calculating (3)
18:        end for
19:      else
20:        for  $s = 1, 2, 3, \dots, S$  do
21:          Update  $\vartheta$  by calling algorithm1( $\vartheta, opt1, B, D, dist, target\_dist, model, loss\_fc$ )
22:          Get  $iou$  by calculating (3)
23:        end for
24:        if  $loss < loss_{min}$  then
25:          Update  $loss_{min}$ 
26:          Reset  $not\_im$ 
27:        else
28:          Update  $not\_im += 1$ 
29:        end if
30:        if  $iou > iou_{max}$  or  $not\_im = 20$  break
31:      end if
32:    end for

```

As it shown in Table 5, the mixed method achieves precision values like the segmented images assays, but the computational cost is higher. Although robustness increased with this methodology, when the switch occurs there is an increase in the loss value, sometimes reaching values even higher than those at the beginning (Fig. 9). This fact could mean that the assay cannot achieve convergence or stay in a local minimum, as happened with the segmented images.

In the modified mixed method, the weights of the network obtained in the first phase are better adapted to the characteristics of the segmented images and this excessive increase in the error is avoided when making the change, as can be seen in the graph in Fig. 11.

The data from the simulations with this second method (Table 6), as in the previous case, reflect levels of precision similar to those obtained with segmented images, and in

this case, the computational cost is also somewhat lower, reaching values of the order of the segmented imaging assays. The latter is mainly due to the fact of avoiding that peak in the realization of the change, thus avoiding back-sliding in learning.

Therefore, this strategy achieves greater robustness without excessively affecting the level of precision or the computational cost. Finally, this methodology was used in a last assay with the non-synthetic dataset overlap to show that it can converge and verify the increase in robustness that occurred.

At first, the assays with this dataset were unable to converge, since when making the first change of distances to segmented, the level of overlap obtained was insufficient and did not converge. For this reason, we decided to carry out a warm up for the learning rate value of the optimizer that works with segmented images, and, in this way, avoid

Table 5 Assays of mixed training method and *C. elegans* images

	Mean error (px)	Min. error (px)	Max. error (px)	IoU	Time (h)
Assay 1	5039	2313	29621	0.85871	4:40:39
Assay 2	5474	1999	29501	0.85436	4:16:11
Assay 3	5154	2457	17990	0.86132	5:11:40

Table 6 Assays with the new modified mixed training method and *C. elegans* images

	Mean error (px)	Min. error (px)	Max. error (px)	IoU	Time (h)
Assay 1	4936	2097	30088	0.86205	4:02:14
Assay 2	5124	2026	18606	0.85914	4:11:54
Assay 3	5279	2346	29410	0.85668	5:13:22

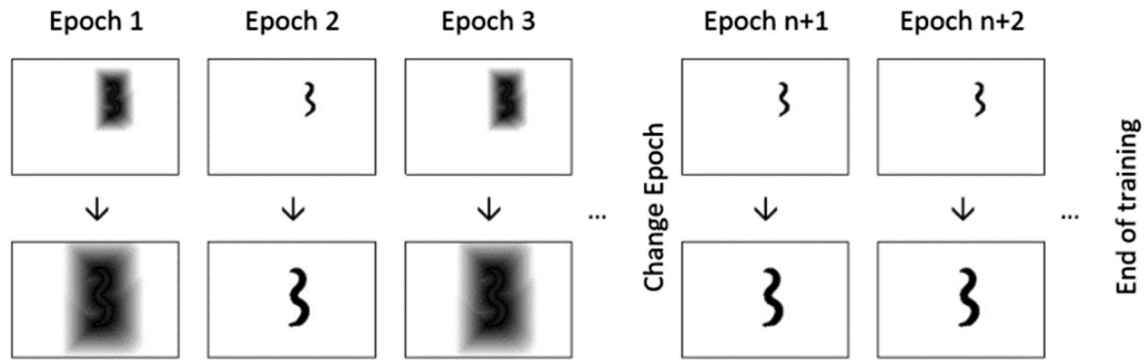


Fig. 9 Schema of modified mixed training method

that in the first segmented time was separated in excess of the solution obtained by the distance-transformed images.

The initial value for the learning rate of the optimizer that worked with segmented images was $2e^{-5}$ and each time it was carried out with segmented images its value increased by $2e^{-5}$ until reaching a value of $1e^{-4}$. In this way, after the first 5 times that this type of image was used, corresponding to the intercalated training phase, this value was reached.

Table 7 shows that this last training method is able to converge with this dataset, reaching a precision level of 0.984 for the IoU metric, representing an increase of 7% with respect to the results with distance-transformed images. Therefore, the increased robustness of this method is demonstrated along with the preservation of the level of precision typical with segmented images.

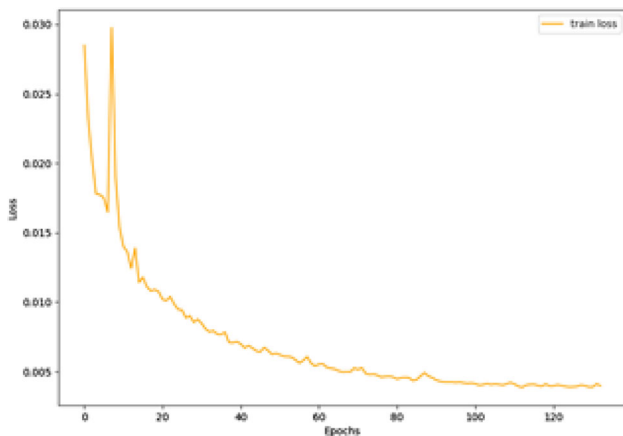


Fig. 10 Downward trend for the assay with the mixed training method

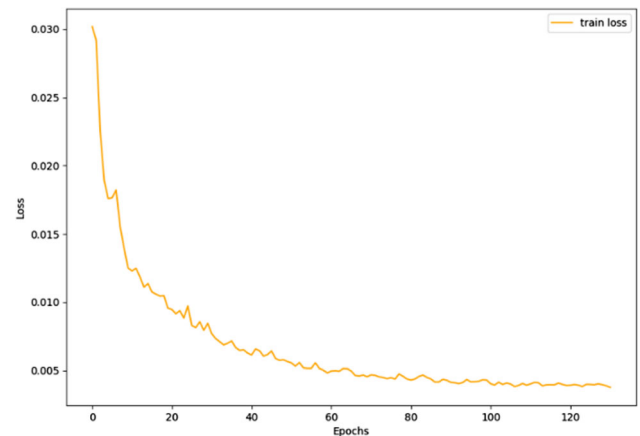


Fig. 11 Downward trend for the assay with the modified mixed training method

Table 7 Assay with new modified mixed training method and no overlap synthetic dataset

	Mean error (px)	Min. error (px)	Max. error (px)	IoU
Assay 1	197	36	434	0.98244
Assay 2	170	29	417	0.98486
Assay 3	222	27	688	0.98039

4 Conclusions

In this paper, a new training method is proposed for those cases in which the percentage of overlap between the two perspectives of the object of interest is small or even null.

It has been demonstrated that the incorporation of distance-transformed images to the training, intercalating these with periods that use segmented images during the first period, solves the correspondence task with greater robustness, thereby preventing convergence problems from occurring during these periods.

Moreover, in addition, all this has been achieved maintaining the level of precision of segmented images and without excessively increasing the computational cost.

Acknowledgements This study was also supported by Plan Nacional de I+D under the project RTI2018-094312-B-I00 and by the European FEDER funds.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Code availability Code is available at <https://github.com/FranciscoNavarroMoya/A-new-training-strategy-for-spatial-transform-nets-STNs.git>.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K (2016) Spatial transformer networks, 2017–2025 [arXiv:1506.02025](https://arxiv.org/abs/1506.02025) [cs.CV]
- Li G, Xu S, Liu X, Li L, Wang C (2018) Jersey number recognition with semi-supervised spatial transformer network. In: 2018 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), pp. 1864–18647. <https://doi.org/10.1109/CVPRW.2018.00231>
- Arcos-García Álvaro, Álvarez-García JA, Soria-Morillo LM (2018) Deep neural network for traffic sign recognition systems: an analysis of spatial transformers and stochastic optimisation methods. *Neural Netw* 99:158–165. <https://doi.org/10.1016/j.neunet.2018.01.005>
- Aubreville M, Krappmann M, Bertram C, Klopffleisch R, Maier A (2017) A guided spatial transformer network for histology cell differentiation. In: Eurographics workshop on visual computing for biology and medicine, pp. 21–25. <https://doi.org/10.2312/vcbm.20171233>
- Lin Y, Wang M, Gu C, Qin J, Bai D, Li J (2018) A cascaded spatial transformer network for oriented equipment detection in thermal images. In: 2018 2nd IEEE conference on energy internet and energy system integration (EI2), pp. 1–5. <https://doi.org/10.1109/EI2.2018.8582248>
- Qian Y, Yang M, Zhao X, Wang C, Wang B (2020) Oriented spatial transformer network for pedestrian detection using fish-eye camera. *IEEE Trans Multimed* 22(2):421–431. <https://doi.org/10.1109/TMM.2019.2929949>
- Zhang X, Gao T, Gao D (2018) A new deep spatial transformer convolutional neural network for image saliency detection. *Des Auto Embed Syst* 22:243–256. <https://doi.org/10.1007/s10617-018-9209-0>
- Fang Y, Zhan B, Cai W, Gao S, Hu B (2019) Locality-constrained spatial transformer network for video crowd counting. In: 2019 IEEE international conference on multimedia and expo (ICME), pp. 814–819. <https://doi.org/10.1109/ICME.2019.00145>
- Li S, Günel S, Ostrek M, Ramdya P, Fua P, Rhodin H (2020) Deformation-aware unpaired image translation for pose estimation on laboratory animals. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp. 13155–13165. <https://doi.org/10.1109/CVPR42600.2020.01317>
- Bhagavatula C, Zhu C, Luu K, Savvides M (2017) Faster than real-time facial alignment: a 3d spatial transformer network approach in unconstrained poses. In: 2017 IEEE international conference on computer vision (ICCV), pp. 4000–4009. <https://doi.org/10.1109/ICCV.2017.429>
- Puchalt JC, Gonzalez-Rojo JF, Gómez-Escribano AP, Vázquez-Manrique RP, Sánchez-Salmerón AJ (2022) Multiview motion tracking based on a cartesian robot to monitor *Caenorhabditis elegans* in standard Petri dishes. *Sci Rep* 12(1):1–11

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.