Deep Decoupling of Defocus and Motion Blur for Dynamic Segmentation SUPPLEMENTARY MATERIAL

Abhijith Punnappurath, Yogesh Balaji, Mahesh Mohan, Rajagopalan A. N.

Department of Electrical Engineering
Indian Institute of Technology Madras, Chennai 600036, India
{ee10d038,ee12b066,ee14d023,raju}@ee.iitm.ac.in

This supplementary material carries additional results and details that could not be provided in the main paper due to space constraints. We first illustrate through a few examples why ours is a more general dynamic segmentation algorithm than our closest competitors, Chakrabarti et al. [4] and Paramanand and Rajagopalan [21]. Next, we provide a visualization of the kernels predicted by our network using the dataset of Kohler et al. [17]. We also perform experiments on the dataset of [17] and on the static scenes from the blur perception dataset [23] which were not included in the main paper. Additional information regarding the implementation of our scheme has also been provided. All references correspond to the main paper unless stated otherwise.

S1 Overview of comparisons with other methods

As mentioned in Section 4 of our main paper, the dynamic segmentation algorithm of [4] is the work most closely related to ours since they also focus on the problem of detecting moving objects from a single blurred image (see caption of Fig. 1 of their paper). However, they segment the moving objects in the scene based on two assumptions – (i) the background is sharp and the foreground object(s) are motion-blurred (see paragraph two, first line in Section 5 of their paper), and (ii) all foreground objects are corrupted by the same kernel (see paragraph one, last line in Section 5). Therefore, their algorithm is only applicable to the very restricted case of a static camera and a single moving object (or multiple spatially non-contiguous objects but all having the same motion). The method of [21], on the other hand, segments a given image into different regions based on the motion; they do not explicitly address the problem of segmenting dynamic objects. However, their algorithm cannot resolve depth-motion ambiguity when segmenting different regions since they do not take depth into account – even in the simple situation of a moving camera imaging a bilayer scene containing a background depth layer and a single stationary object in the foreground depth layer, [21] may incorrectly label the object as dynamic since the blur incurred by the foreground layer will be different from the background due to the difference in depth. See Table S1 for a succinct overview of the methods in [4] and [21], and D^{3}M.

Table S1. A comparison of [4], [21] and D^{3}M. Y = Yes, N = No, and $\times$ = don't care condition (can be either Y or N).

S.No.	Sharp object	Sharp back	Single object	Single depth	[4]	[21]	D^{3}M	Remarks
1	Y	Y	$\times$	$\times$	✓	✓	✓	Static camera and scene.
2	N	Y	Y	Y	✓	✓	✓	All three methods work.
3	$\times$	N	Y	Y	✗	✓	✓	Violates [4]'s focused background assumption.
4	N	$\times$	N	Y	✗	✓	✓	Violates [4]'s single object assumption.
5	Y	N	Y	N	✗	✗	✓	Violates [4]'s focused background assumption. [21] may flag foreground object as dynamic even if background is simply out-of-focus.
6	N	Y	Y	N	✗	✗	✓	[4] and [21] may flag stationary defocused foreground object as dynamic.
7	N	Y	N	N	✗	✗	✓	Violates [4]'s single object assumption. [21] may flag stationary defocused foreground object as dynamic.
8	N	N	$\times$	N	✗	✗	✓	Violates [4]'s focused background assumption. [21] cannot resolve depth-motion ambiguity.

A synthetic example demonstrating four different interesting conditions (rows 2, 3, and 8 of Table S1) is shown in Fig. S1. The first row of Fig. S1 shows the case of a static camera and a moving object with a single depth layer, and it can be seen that all the three methods correctly identify the object as dynamic. The second row depicts the case where the camera is panning at a speed that matches the velocity of the dynamic object such that, in the resultant image, the object appears sharp but the background is blurred, while the third row demonstrates the practically common situation when the hand-held camera is not intentionally tracking the dynamic object, but merely observing a dynamic scene resulting in the entire image being blurred. In both these cases, the assumption made by Chakrabarti et al. [4] of a sharp background and a motion-blurred object is violated leading to an incorrect segmentation. The last row shows a bilayer scene where only the camera is in motion while the object in the foreground depth layer is stationary. In this case, [21] wrongly classifies the object as dynamic because the kernels on the foreground object do not match the ones on the background because of depth differences. Observe that our algorithm correctly identifies the object as being dynamic in the first three cases and static in the fourth. This is due to the fact that our inference of whether or not an object is dynamic is independent of the camera, and is measured with respect to the background. Furthermore, we take depth into account to ascertain whether the object is stationary or moving.

We would also like to add that we have not compared our method against Shi et al. [23] although we have used images from their dataset because their technique is oriented towards detecting which pixels in an image are blurred

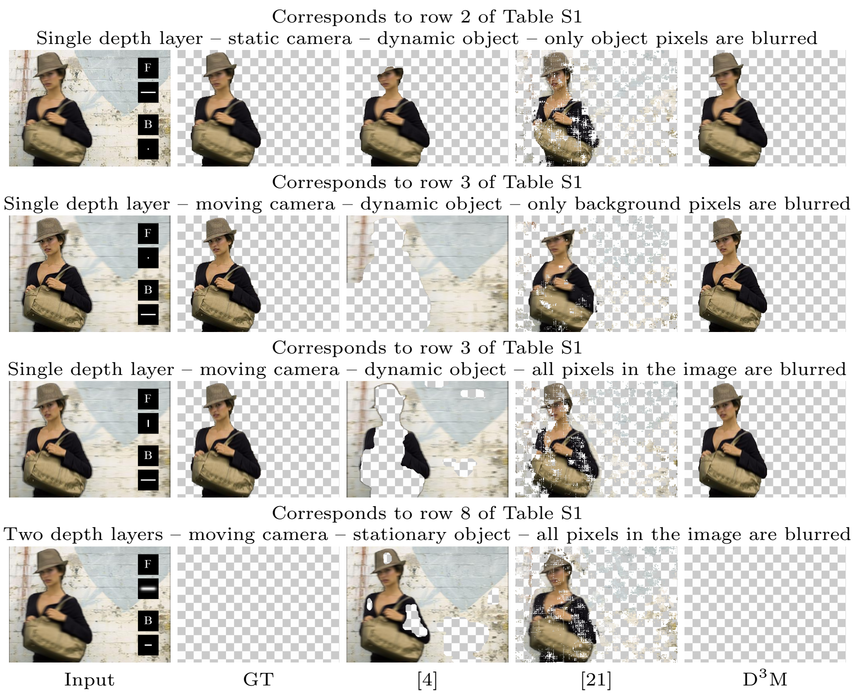
Corresponds to row 2 of Table S1
Single depth layer – static camera – dynamic object – only object pixels are blurred
Corresponds to row 3 of Table S1
Single depth layer – moving camera – dynamic object – only background pixels are blurred
Corresponds to row 3 of Table S1
Single depth layer – moving camera – dynamic object – all pixels in the image are blurred
Corresponds to row 8 of Table S1
Two depth layers – moving camera – stationary object – all pixels in the image are blurred
F
B
Input
GT
[4]
[21]
D^{3}M

Fig. S1. A synthetic example demonstrating D^{3}M's ability to segment moving objects under various types of camera and object motions. The foreground and background kernels are overlaid on the input images. See the text for details.

and which are not. In fact, there are several methods ((i) Liu et al., "Image partial blur detection and classification," in CVPR 2008, (ii) Su et al., "Blurred image region detection and classification", in ACM international conference on Multimedia, 2011, (iii) Pang et al., "Classifying discriminative features for blur detection," IEEE Transactions on Cybernetics, 2015, and (iv) Lee and Kim, "Blurred image region detection and segmentation," in ICIP, 2014) that exclusively tackle the problem of detecting blurred pixels in natural images. However, none of these techniques address the issue of segmenting dynamic objects, and that is why these comparisons have been omitted. On the other hand, our proposed framework is end-to-end and can unambiguously detect moving objects at different depth layers directly from the input blurred observation.

S2 Network assessment using Kohler et al. [17] dataset

As discussed in Section 2.1 of our main paper, we evaluate our network's kernel prediction accuracy by comparing the ground truth kernels in [17] with the PSFs predicted by our network using the cross-correlation metric. An input blurred image from the dataset of [17] is shown in Fig. S2(a). Fig. S2(b) shows the ground truth kernels for the blurred image in Fig. S2(a). Note that these kernels were recorded from *real* camera motion. The PSFs predicted by our network are displayed in Fig. S2(c), and it can be observed that our CNN outputs a close approximation to the actual kernels.

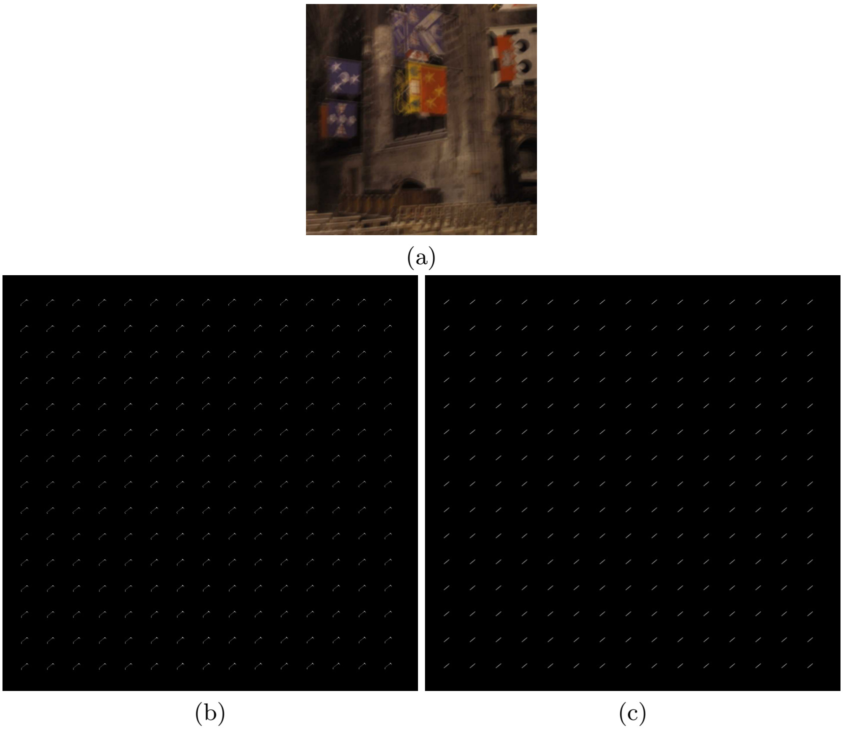

Fig. S2. (a) Input blurred image, (b) ground truth kernels, and (c) output kernels of our CNN. (Kernels are best viewed as PDF.)

S3 Additional results on the datasets of [17] and [23]

Due to space constraints, we did not include segmentation results of our algorithm on the dataset of Kohler et al. [17] in our main paper. While all the scenes in this dataset are static, this is nevertheless relevant from a segmentation perspective; our algorithm is expected to classify all pixels as stationary in all images. For quantitative evaluation of our dynamic segmentation results, we compute specificity (SPC) as

$$\mathrm{SPC} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{1}$$

where TN and FP denote true negative and false positive, respectively, and 'positive' is when a pixel is classified as being dynamic[1]. The denominator in equation (1) is equal to the total number of pixels in the image, while the numerator corresponds to the number of pixels classified as static. The SPC values averaged over all 48 images using the methods in [4] and [21], and D^3M, are provided in Table S2. Our method records fewer false positives than the two competing techniques, and rarely flags a static pixel as moving.

From the blur perception dataset of [23], we had reported (in our main paper) quantitative results only on the 296 *dynamic* scenes. We had included just two static examples in row two of Fig. 6 in the main paper. For a comprehensive quantitative assessment, we computed the average SPC value using the remaining 704 static images, and these are reported in Table S2. It can be seen from the results that D^3M yet again performs better than others.

Table S2. Average SPC values for the methods in [4] and [21], and D^3M, on the dataset of Kohler et al. [17] and the static scenes from the blur perception dataset [23]. (Higher SPC is better.)

	[4]	[21]	D^3M
Kohler et al. [17]	0.596	0.544	0.939
Blur perception dataset [23]	0.741	0.675	0.964

S4 Additional experimental details

We used the *czf_segment* code downloaded from the webpage of the first author of [4] to report comparisons with their method. This code combines their proposed blur cue and the color information in the image using an MRF model to output a hard segmentation of the moving objects (see paragraph two of Section 7 of [4]). Similar to [4], our segmentations are also hard. So also is the output of [21]. Hence, we computed, for each of these three methods – [4], [21] and D^3M, a single value of precision and a single value of recall for each input image. And the average precision and recall values over all images were reported in Table 1 of our main paper.

For quantitative evaluation on the *dynamic scenes* in the blur perception dataset [23], in Section 4.2 of our main paper, we made use of the ground truth masks provided by the authors of [23]. However, as already mentioned in our main paper, this dataset is originally designed for benchmarking of algorithms that detect blurred pixels in an image whereas our objective is dynamic segmentation. Hence, for the 296 dynamic scenes from the dataset that we used for our

[1] We compute specificity and not precision and recall for these cases since there are no true positives and false negatives for static scenes i.e., for images with no dynamic pixels in the ground truth, precision is zero or undefined, while recall is undefined.

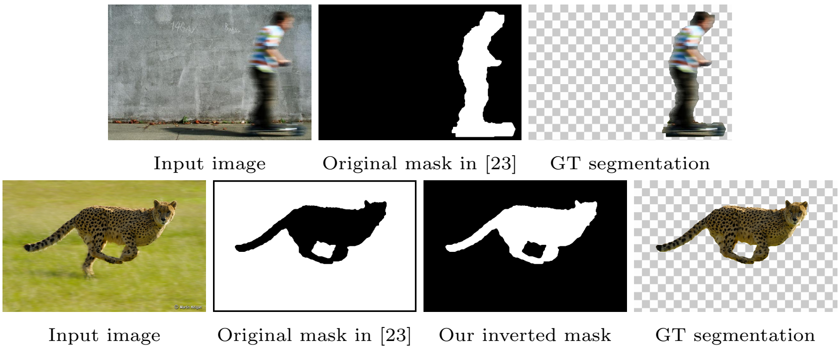

Fig. S3. An example depicting how the ground truth masks provided by the authors of [23] should be interpreted for the dynamic segmentation task at hand.

study, the ground truth masks as given by [23] do not always directly correspond with the dynamic objects. This is illustrated in Fig. S3. For the example in row one, we directly use the mask provided in [23] since it correctly maps to the dynamic pixels in the scene. Observe that the foreground pixels are blurred while the background is sharp i.e., foreground pixels are flagged one while the background pixels are marked as zero. However, for the example in row two where the camera is panning with the foreground object, the background is blurred while the foreground pixels are not. In this case, it is the inverted mask that corresponds to the dynamic foreground object. Therefore, for our experiments, we manually inverted the mask for all such images in the dataset of [23] where the dynamic foreground object is sharp and the background is blurred.

We used the following empirically-determined values for the weighting constants in all our experiments: $\alpha = 0.8$, $\beta_1 = 100$, $\beta_2 = 50$ for graphcut (see page six, last paragraph of our main paper). We used four PSFs (i.e, $N = 4$) from the background to estimate ω_0 (see page 10, third paragraph of our main paper). In Section 3.2 of our main paper (page 10, third paragraph), we used a threshold value of 0.5 on cross-correlation to check if the estimated PSF is consistent with the kernel predicted by our network, and ascertain whether the pixel under consideration belongs to a dynamic object. Our MATLAB implementation of D^{3}M takes roughly 4 minutes to classify the composite kernels and perform segmentation on an image of size 640×480 pixels.