Deep Decoupling of Defocus and Motion Blur for Dynamic Segmentation

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

Abstract. We address the challenging problem of segmenting dynamic objects given a single space-variantly blurred image of a 3D scene captured using a hand-held camera. The blur induced at a particular pixel on a moving object is due to the combined effects of camera motion, the object's own independent motion during exposure, its relative depth in the scene, and defocusing due to lens settings. We develop a deep convolutional neural network (CNN) to predict the probabilistic distribution of the composite kernel which is the convolution of motion blur and defocus kernels at each pixel. Based on the defocus component, we segment the image into different depth layers. We then judiciously exploit the motion component present in the composite kernels to automatically segment dynamic objects at each depth layer. Jointly handling defocus and motion blur enables us to resolve depth-motion ambiguity which has been a major limitation of the existing segmentation algorithms. Experimental evaluations on synthetic and real data reveal that our method significantly outperforms contemporary techniques.

Keywords: Segmentation, neural network, defocus blur, motion blur

1 Introduction

Segmentation of dynamic objects in a scene is a widely researched problem as it forms the first step for many image processing and computer vision applications such as surveillance, action recognition, scene understanding etc. Classical video-based motion segmentation algorithms [22, 8] assume that the camera is stationary and only the object of interest is in motion in the scene, thus allowing them to learn the static background and separate out the dynamic object. However, the assumption of a static camera does not hold in most real-world applications – the camera might be hand-held or mounted on a moving vehicle and there could be significant parallax effects due to the 3D nature of the scene. The combination of a moving camera and a dynamic 3D scene often introduces an additional challenge in the form of blurring. To bring the entire 3D scene into focus, one must select a small aperture (large depth of field), and thereby a larger exposure time. But this increases the chances of motion blur since both object and camera are in motion. On the other hand, reducing the exposure time

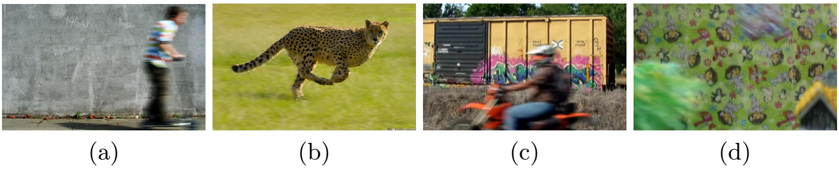

Fig. 1. Dynamic scenes. (a-b) Blur perception dataset [23], (c) a frame extracted from a video downloaded from the internet, and (d) an indoor image we captured ourselves using a hand-held camera.

by choosing a large aperture (small depth of field) results in depth dependent defocus blur. Thus, there exists a trade-off between defocus and motion blur, and it is difficult to completely avoid both in the case of a dynamic 3D scene. It is important to note that both kinds of blur degrade the performance of conventional segmentation algorithms that rely on feature correspondences between video frames to extract the moving objects.

Although blur has traditionally been regarded as an undesirable effect, works exist [11, 7] that have used motion blur itself as a cue for segmentation, while others [5, 10] have exploited defocus blur as a cue to estimate depth. While Favaro and Soatto [11] addressed the problem of dynamic scene segmentation by restricting the motion of the object to pure in-plane translations, Deng et al. [7] allowed for non-uniform object motion. However, neither of these works [11, 7] consider defocus blur. Classical depth from defocus (DFD) algorithms [5, 10] assume a stationary camera and a static scene, and use multiple images captured under different lens settings to recover the depth map. The DFD technique in [25] allows for object and camera motion *between* frames. However, they do not analyse the effect of motion blur *within* a frame. Paramanand and Rajagopalan [20] extend the DFD framework to the case where the camera is free to undergo in-plane translations during image capture, but the scene is constrained to be static. It is also important to note that all of the above methods require two or more images as input.

The problem of motion segmentation becomes far more ill-posed if only a *single* blurred image of the scene is available. Fig. 1 shows four real examples of dynamic scenes. Chakrabarti et al. [4] tackle this situation by assuming a static camera and restricting the motion of the object be either horizontal or vertical. However, they ignore the effects of defocus. Paramanand and Rajagopalan [21] too ignore defocus and present an approach to segment a blurred image captured using a hand-held camera into different regions based on the motion kernels, but they do not determine whether an object is moving or not.

There are inherent ambiguities in segmenting moving objects from a single image of a 3D scene blurred due to camera shake if the defocus effect is ignored. This is because a fast-moving object which is relatively far away from the camera can cause the same blurring effect as a stationary object close to the camera. Existing motion segmentation works such as [4, 21] cannot resolve this issue. This depth-motion ambiguity exists because the depth map of the scene is unknown. Interestingly, defocus blur can be used to recover the depth map even from a single image. In fact, recovering the depth map from a single image is a research topic in its own right, and several works [24, 32] have addressed this issue. But these works assume a static camera and scene.

It is clear from the preceding discussions that motion and defocus blurs should be considered jointly to obtain an unambiguous classification of the dynamic objects when only a single blurred image of the 3D scene is available. However, the interplay of these two blurs has not been explored in the context of motion segmentation, and therein lies the main novelty of our work. It is noteworthy that only a few works [4, 21] have addressed the challenging problem of motion segmentation from a single image despite the fact that it is a problem of high contemporary relevance.

Convolutional neural networks (CNN) have been successfully employed in recent times to classify kernels [27, 31, 1]. However, these works assume that the input image is corrupted either by defocus or motion blur, but not both. In this work, we propose a new CNN architecture to classify the composite kernel obtained as the convolution of motion and defocus kernels. Such a formulation allows us to decouple the defocus and motion components at each pixel in the image based on the combined kernel. We demonstrate how the defocus kernels reveal the depth map, while the motion kernels can be harnessed to segregate the dynamic objects in the scene. Our proposed scheme results in a natural confluence of deep learning, depth from defocus, and motion blur to solve a challenging problem. While on the one hand, our method generalizes to handling motion and defocus blur, it is these two blurs themselves that enable us to resolve motion and depth ambiguity.

Contributions

1. To the best of our knowledge, this is the first attempt at segmenting moving objects given a single non-uniformly blurred image of a 3D scene.
2. We propose a CNN architecture to predict the probabilistic distribution of the composite kernel resulting from optical defocus and non-uniform motion blur.
3. Our joint model for defocus and motion blur enables us to overcome depth-motion ambiguity which existing segmentation works cannot resolve.

2 Kernel classification using CNN

Consider a dynamic 3D scene captured by an out-of-focus moving camera with no restrictions imposed on the camera or object motion during exposure. Then, the point spread function (PSF) or the kernel observed at a pixel corresponding to a planar region in the scene can be modeled as the convolution of a defocus PSF and a motion PSF [20] i.e., the composite kernel h_c at a pixel $(i, j) \in \Gamma$ can be expressed as $h_c(i, j) = h_d(i, j) * h_m(i, j)$, where h_d and h_m represent the defocus and motion kernels, respectively, * denotes convolution, and Γ is the discrete 2D image-coordinate grid. Note that for a pixel (i, j) lying on a moving object, the motion kernel h_m is the net result of camera as well as object motion. Motivated by this fact, we propose to estimate the spatially-varying composite kernel at a patch level using a CNN. Following [27], we approximate the motion PSF h_m at any given pixel (i, j) by a linear kernel which is parameterized by its length l and its orientation ϕ i.e., $h_m(i, j) \approx \Psi_{1_{(i,j)}}(l, \phi)$. Since our objective in this work is only to segment the scene, small errors that may arise from such an approximation do not seriously hinder our ability to localize the dynamic objects. It is to be noted that [27] use this approximation to tackle the deblurring problem where the requirement of accurate PSFs is more stringent. The defocus PSF h_d, which we approximate by a Gaussian following DFD algorithms [12], is characterized by the standard deviation of the Gaussian and can be expressed as $h_d(i, j) = \Psi_{2_{(i,j)}}(\sigma)$. We can use our CNN to predict as well as decompose the composite kernel at each pixel by dividing the image into overlapping patches. We use a patch size of 30×30 pixels and assume that within a patch, the blur is space-invariant. We work on grayscale images because the blur incurred by all the three color channels is the same. Note that the composite kernel h_c is parameterized by l, ϕ and σ alone. The advantage of such a formulation is that, aided by the defocus component σ which yields the depth layers in the scene, we can unambiguously ascertain the presence of moving objects (as we shall see in Section 3) using the motion component characterized by l and ϕ. We would like to highlight once again that existing neural network-based works [27, 31, 1] have only considered the two blurs in isolation, and this is the first time a CNN architecture is being proposed to classify the combined defocus and motion PSF.

We discretize the parameter space (l, ϕ, σ) in the following manner. For motion length, we choose seven values $l = 1$ to 13 pixels in increments of 2 pixels, while for motion orientation, we select six samples $\phi = 0°$ to $150°$ in intervals of $30°$. This gives us a total of 37 motion PSFs since the kernel is an impulse for $l = 1$ irrespective of ϕ. By choosing 11 discrete values for standard deviation as $\sigma = 0$ to 2 in increments of 0.2, we obtain a total of $37 \times 11 = 407$ composite kernels which form the candidate set on which we train our neural network.

The complete architecture of our CNN is shown in Fig. 2. There are eight layers. The first, third and fifth are convolutional layers employing (3×3) filters,

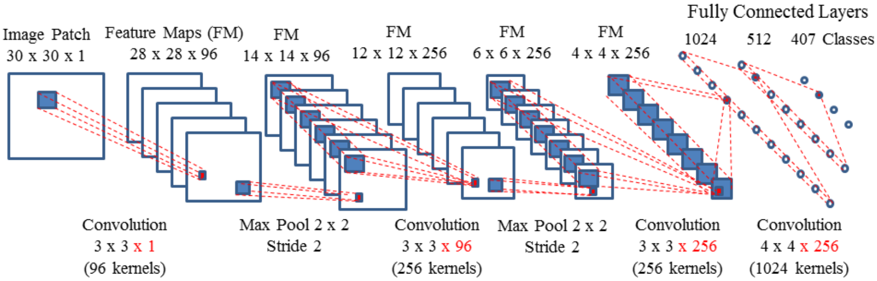

Fig. 2. Our CNN architecture for predicting the composite kernels.

and all three are followed by ReLU non-linear transform. The second and fourth are max-pooling layers over 2×2 cells with a stride of 2. The sixth and seventh are fully connected layers with 1024 and 512 neurons, respectively, while the final eighth layer is a soft-max layer with 407 labels corresponding to the composite kernels in the candidate set.

We selected 10,000 images at random from the PASCAL VOC 2012 dataset [9] for training our CNN. For each training image, we first generated a synthetically blurred image by convolving it with a composite PSF from the candidate kernel set, and then cropped a random 30×30 patch from the blurred image. The patch was then converted to grayscale. This process was repeated for all the training images and for all the composite kernels giving us approximately 4 million patches. We performed data augmentation on the training set by rotating the images by 45°. The resultant 8 million blurred patches and their corresponding ground truth labels were used to train our CNN. The training was done on MatConvNet [29] using stochastic gradient descent with a batch size of 400.

The quantization of ϕ of our trained network is in intervals of 30° which is too coarse, and insufficient for the segmentation task at hand. Hence, we propose to improve the angular resolution based on the observation in [27] that if the CNN predicts the orientation of a patch which is rotated about its center by an angle ϕ_1 as ϕ, then the orientation of the kernel corresponding to the original patch is simply $\phi - \phi_1$. Note that rotation has no effect on the defocus kernel since it is centrally symmetric. Thus, by feeding rotated patches to the network, we can extend the candidate set without retraining the CNN. We choose $\phi_1 \in \{-10°, -20°\}$ so that our motion orientation set expands to $\{0°, 10°, 20°, ..., 170°\}$. This means that our CNN can now totally predict ($109 \times 11 =$) 1199 composite kernels; almost 3 times the original number!

We demonstrate, with a synthetic example, how the composite kernels predicted by our network are decomposed into the corresponding motion and defocus PSFs. Fig. 3(a) shows a blurred image from outside our training set which is provided as input to our network. For ease of visual discrimination of the kernels in various depth layers/moving objects, in this example, we have selected space-invariant camera motion. So also is the motion of each object. However, we would like to point out that our segmentation algorithm (to be discussed in detail in Section 3) is equally equipped to handle non-uniform camera and object motion, and we have considered such examples in our real experiments in Section 4. Note that, for visualization, we have shown kernels only at uniformly-spaced sparse locations in Figs. 3(b) and (d), although our CNN predicts a kernel at each pixel. Observe that there are three depth layers (see Fig. 3(c)). The trees and the two moving aircraft comprise the depth layer farthest from the camera. We have selected the motion of the two birds such that the one in the middle depth layer is stationary with respect to the background region i.e., the trees, while the one in the layer closest to the camera is moving. The composite kernels predicted by our CNN are shown in Fig. 3(e). These can be decomposed into their respective defocus and motion kernels. Since each defocus kernel h_d is associated with a unique σ, we can directly obtain a layered depth map of

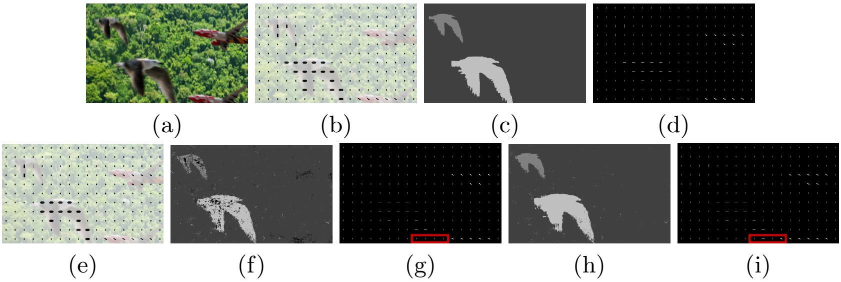

Fig. 3. A synthetic example demonstrating our CNN's ability to predict the composite kernels. (a) Blurred input image, (b) ground truth composite kernels, (c) ground truth depth map, (d) ground truth motion kernels (without considering defocus blur), (e) composite kernels predicted by our CNN, (f) depth map decomposed from (e), (g) motion PSFs decomposed from (e), (h) refined depth map after graphcut, and (i) refined motion kernels after graphcut. By convention, a scene point that is closer to the camera has a higher intensity value in the depth map than one that is farther away.

the scene from this blur component as follows. We know that the blur radius r_b, which is linearly related to σ, is given by [5]

$$r_b = r_o v_o \left(\frac{1}{F_l} - \frac{1}{v_o} - \frac{1}{d} \right) = r_o v_o \left(\frac{1}{u} - \frac{1}{d} \right) \tag{1}$$

where v_o is the distance between the lens and the image plane, and r_o, F_l and u denote aperture radius, focal length and working distance, respectively. These parameters are typically known for a calibrated camera. The depth is denoted by d. The depth map and the motion PSFs are shown in Figs. 3(f) and (g).

Since the label at each pixel is being predicted independently, we are not taking advantage of the prior knowledge that the depth map is homogeneous in most regions with abrupt changes only at the borders of the depth layers. We enforce this condition by solving an MRF optimization problem using the probabilities computed at the final soft-max layer of our CNN. From 1199 labels, we pick the maximum probability corresponding to each of the 11 σ values, and pass it to an off-the-shelf graphcut algorithm [2]. We define the pairwise cost as $\beta_1 \times (1 - \alpha^{|\sigma_a - \sigma_b|})$, where $|\cdot|$ denotes the absolute value, and β_1, α are positive scalars. Likewise, nearby pixels should have similar motion PSFs since the camera and objects typically move smoothly during exposure. For motion kernels, we define the pairwise cost for graphcut as $\beta_2 \times [(l_a \cos(\phi_a) - l_b \cos(\phi_b))^2 + (l_a \sin(\phi_a) - l_b \sin(\phi_b))^2]$, where β_2 is a positive weighting constant. The refined depth map and motion PSFs after graphcut are shown in Figs. 3(h) and (i), respectively. Observe that some of the errors in the depth map and the motion kernels (compare the boxes in red in Figs. 3(g) and (i)) are reduced after applying graphcut.

2.1 Network assessment

As discussed in Section 2, we approximate the arbitrarily-shaped PSF at each pixel resulting from real camera and/or object motion by a linear motion kernel. To compute the kernel prediction accuracy of our network, we use the dataset in [17] which, to our knowledge, is the only dataset with space-varying blur for which the ground truth PSF at each pixel is available. The dataset contains 48 blurred images of static scenes, and is designed for the benchmarking of single image blind deblurring algorithms. To compute prediction accuracy, we compare at each pixel the ground truth kernel and the PSF predicted by our network using cross-correlation [15], which is a standard metric for kernel similarity. The normalized cross-correlation measure varies between zero and one, and a value of 0.6 or greater indicates a good match [15]. We obtained a cross-correlation value of 0.636 averaged over all pixels and all 48 images indicating that our network generalizes quite effectively to data outside the training set. See the supplementary material for more details.

3 Scene segmentation

In this section, we describe in detail the steps involved in segmenting the moving objects at different depth layers based on the defocus and motion PSFs decoupled from the composite kernels predicted by our CNN. We make the following assumptions. When an object appears defocus blurred, it can be on either side of the focal plane. To circumvent this ambiguity, following other DFD works [32], we also assume that the defocus blur varies monotonically (either increases or decreases) from foreground to background. We also assume that the depth layers in the scene are predominantly fronto-parallel and planar.

3.1 Segmenting moving objects in the reference depth layer

We label the depth layer with the maximum area in the depth map as the reference depth layer d_0. For the example we had considered earlier in Fig. 3, the reference depth layer contains the trees and the two aircraft as can be observed from the depth map in Fig. 3(h). Let $\Gamma_0(\subset \Gamma)$ denote the set of pixels in the image at this reference depth d_0. It is also reasonable to assume that d_0 houses the background region (the trees). Our objective is to separate out the background region from the dynamic objects (the two moving aircraft) at depth d_0. To accomplish this, we exploit the fact that the blur observed on the background region due to the motion of the camera will be different from the blur induced on a moving object which experiences the combined effects of object and camera motion (see Fig. 3(i)). We use the blur compatibility criterion defined in [21] to compare two motion PSFs, and decide whether they correspond to the same or different motion.

Since we are initially concerned only with the depth layer d_0, we consider a static fronto-parallel planar scene and briefly review the relationship between the space-varying PSF and the global camera motion using a non-uniform motion blur model. The blurred image g can be modeled as a weighted average of warped instances of the latent image f [30,13,14,28]. Mathematically, $g = \sum_{k=1}^{|\mathbf{T}|} \omega_0(k) f_k$, where $\mathbf{T}$ is the discrete set of transformations that the camera is free to undergo, and $|\cdot|$ denotes cardinality. f_k is a warped instance of the latent image obtained by transforming the 2D image grid $\mathit{\Gamma}$ on which f is defined using the homography $\mathbf{H_k}$. The parameter ω_0 depicts the camera motion i.e., for each transformation $k \in \mathbf{T}$, the value of $\omega_0(k)$ denotes the fraction of the total exposure duration for which the camera stayed in the position that caused the transformation $\mathbf{H_k}$ on the image coordinates. The blurred image can be equivalently modeled with a space-variant PSF h as [26]

$$g(i,j) = f *_v h(i,j) = \sum_{m,n} f(i-m, j-n) \times h(i-m, j-n; m, n) \tag{2}$$

where $h(i,j,;)$ denotes the PSF at the pixel (i,j), and $*_v$ represents the space-varying blurring operation. The PSF can be expressed in terms of the weights ω_0 as

$$h(i,j;m,n) = \sum_{k=1}^{|\mathbf{T}|} \omega_0(k) \times \delta(m - (i_k - i), n - (j_k - j)) \tag{3}$$

where δ indicates the 2D Kronecker delta, and (i_k, j_k) denotes the transformed image coordinates when $\mathbf{H_k}^{-1}$ is applied on the pixel (i,j).

If the camera intrinsics are fixed, the homography $\mathbf{H_k}$ has six degrees of freedom arising from the translations along and rotations about the three axes [17]. However, it has been pointed out in [13,14] that in most practical scenarios, we can model the camera trajectory using just in-plane translations and rotations. The homography $\mathbf{H_k}$ is then parameterized by t_{x_k} and t_{y_k} which represent the translations along X and Y axes, respectively, and θ_{z_k} which represents the rotation about the Z axis.

Our aim is to determine the set of pixels $\mathit{\Gamma}_{0_b}$ $(\subset \mathit{\Gamma}_0)$ at the reference depth d_0 which belong to the background. For the sake of discussion, let us assume we know a particular pixel $(i,j) \in \mathit{\Gamma}_{0_b}$. We relax this assumption later. Let h_{m_1} denote the motion PSF at this particular pixel. Let the set of transformations from the discretized motion space $\mathbf{T}$ (on which ω_0 is defined) that shift the pixel (i,j) to a location where h_{m_1} has a positive entry be denoted by $\boldsymbol{\mathcal{T}_1} = \{k : h_{m_1}(i,j; i_k - i, j_k - j) > 0\}$. To check whether a PSF h_{m_2} at another pixel from the set $\mathit{\Gamma}_0$ belongs to the background region, we intersect its transformation support $\boldsymbol{\mathcal{T}_2}$ with $\boldsymbol{\mathcal{T}_1}$ to get the common transformation space $\boldsymbol{\mathcal{T}_{12}} = \boldsymbol{\mathcal{T}_1} \cap \boldsymbol{\mathcal{T}_2}$. We then regenerate the kernels $\hat{h}_{m_1}$ and $\hat{h}_{m_2}$ using $\boldsymbol{\mathcal{T}_{12}}$, and verify whether the actual PSFs have positive entries at locations other than those in the regenerated kernels. If the number of such entries is above a threshold, we can conclude that h_{m_2} corresponds to a PSF on a moving object since there are no common transformations between $\boldsymbol{\mathcal{T}_1}$ and $\boldsymbol{\mathcal{T}_2}$ that can correctly reconstruct both h_{m_1} and h_{m_2} i.e., the two kernels h_{m_1} and h_{m_2} are not compatible.

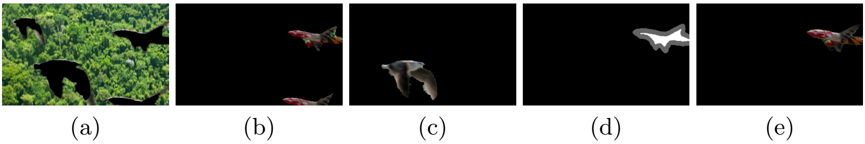
(a) (b) (c) (d) (e)

Fig. 4. Motion segmentation results for the example in Fig. 3. (a) Segmented background at reference depth d_0 (remaining portions have been blacked out), (b) dynamic objects at d_0, (c) dynamic object in the foreground depth layer, (d) automatically generated trimap for the aircraft on the top-right from our segmentation output in (b), and (e) refined borders after matting.

To automatically segment the background region (i.e., obtain Γ_{0_b} from Γ_0) *without* the knowledge of a reference pixel belonging to Γ_{0_b}, we assume that in the reference depth layer d_0, the background occupies a larger area as compared to the moving objects (the trees cover more pixels than the two aircraft). To classify the background pixels, we randomly pick a location (i, j) from Γ_0, and test the compatibility of the PSF at this point with the PSFs at all the remaining points in Γ_0. This gives us a set of kernels that are compatible with the PSF at (i, j). If the cardinality of this set is greater than half the cardinality of the set Γ_0, we label the points in this set as the background region Γ_{0_b}. If this condition is not met, we pick another random location from Γ_0, and repeat the above steps until convergence. Note that the remaining pixels $\Gamma_{0_{mov}} = \Gamma_0 \setminus \Gamma_{0_b}$ automatically reveal the moving objects in d_0 (here $\setminus$ denotes set difference). The segmented background and dynamic objects in d_0 for the example in Fig. 3 are shown in Figs. 4(a) and (b), respectively. Observe that the trees which correspond to the background have been separated out from the moving aircraft. Also note that the dynamic objects need not be spatially contiguous.

3.2 Segmenting moving objects at other depths

Having segmented the background and the moving objects in the reference depth layer, we proceed to the other depth layers (the two foreground layers containing the two birds as can be seen from our depth map in Fig. 3(h)). In this section, we discuss how the motion PSF at a pixel (i, j) at a different depth d_p can be determined if ω_0, which denotes the camera motion experienced by the background at the reference depth d_0, is known. If this estimated PSF is not 'consistent' with the motion PSF predicted by our CNN at (i, j), then we can conclude that there is a moving object at that location.

We first examine the relationship between ω_0 and the motion PSF at a different depth layer. Consider a static 3D scene imaged by a moving camera. The PSF at each pixel now also varies as a function of the depth. However, since the camera motion is the same for all the image points, it is possible to determine the PSF at any depth layer if ω_0 (parameterized by $t_{x_k}, t_{y_k}, \theta_{z_k}$) at the reference depth layer d_0, and the depth map are known. We can express the transformation undergone by a point at a different depth d_p in terms of a scale factor $s_p = \frac{d_p}{d_0}$, which is the relative depth, and the parameters of the homography $\mathbf{H_k}$ as $t_{x_{k_p}} = \frac{t_{x_k}}{s_p}$, $t_{y_{k_p}} = \frac{t_{y_k}}{s_p}$. The rotation parameter θ_{z_k} is not affected by depth, and only the translation parameters get scaled according to the depth value.

Let $\mathbf{H}_{k_p}$ denote the transformation with the parameters $t_{x_{k_p}}$, $t_{y_{k_p}}$ and θ_{z_k}. Then the PSF at a pixel (i, j) can be expressed in the same manner as equation (3) with (i_k, j_k) replaced by (i_{k_p}, j_{k_p}), where (i_{k_p}, j_{k_p}) is obtained by applying $\mathbf{H}_{k_p}^{-1}$ on (i, j). This is to say that the blurred image of the 3D scene can be related to the latent image f through the space variant blurring operation in equation (2) wherein the PSF h depends on the camera motion ω_0 and the depth d_p.

We pick N points at random from the background region Γ_{0_b}. Next, we estimate the camera motion ω_0 using the motion PSFs predicted by our CNN at these locations. For this purpose, we follow the procedure in Section 3.2 of [21] which explains how space-varying camera motion can be estimated from a set of motion kernels. Once ω_0 has been estimated, the next step is to determine the motion PSFs at other depth layers by scaling the translational parameters. This requires the knowledge of the scale factor $s_p = \frac{d_p}{d_0}$. From equation (1), it can be seen that σ_0 at a reference depth d_0 is related to σ_p at a different depth d_p by a scale factor $\frac{\left(\frac{1}{u} - \frac{1}{d_0}\right)}{\left(\frac{1}{u} - \frac{1}{d_p}\right)}$. Since σ_0 and σ_p are known from the defocus component of the predicted composite kernel, s_p can be determined, and the motion PSFs $\hat{h}_{m_p}$ at all other depth layers can be estimated by scaling the translations. We compare the estimated PSF $\hat{h}_{m_p}$ with the motion PSF predicted by our CNN h_{m_p} using cross-correlation [15]. If the normalized cross-correlation value is below a certain threshold, we declare that there is a moving object at that location. Thus, at each depth layer $d_p, p \neq 0$, we may obtain a set of pixels that are inconsistent with the motion of the camera, and these pixels reveal the moving objects. See Fig. 4(c). Our algorithm rightly declares that there are no moving objects in the middle layer. On the other hand, the bird in the depth layer closest to the camera has a different motion and is correctly classified as dynamic.

3.3 Refining borders

Since we adopt a patch-based approach, the composite kernel predicted by our CNN can be erroneous at the boundaries of the moving objects. To obtain an accurate demarcation, we use the closed-form matting algorithm in [18] which has been successfully applied on blurred images [16, 3, 6]. Note that we can generate the trimap, which must also be provided as input to their algorithm, automatically without any user intervention. This is illustrated through an example in Fig. 4. In Fig. 4(d), observe that we have shrunk the region labeled as a moving object after motion segmentation, and flagged the pixels lying within it as sure foreground, while the pixels lying outside the expanded region have been flagged as sure background. It can be seen from Fig. 4(e) that the object boundaries are accurately recovered post matting. An overview of our proposed scheme, which we abbreviate as D^{3}M, is outlined in Algorithm 1.

Algorithm 1 D^{3}M: Deep Decoupling of Defocus and Motion blur for dynamic segmentation

```
Input: Single blurred observation.
Output: A segmentation of the moving objects in the scene.
Decompose the image into overlapping patches of size $30 \times 30$ pixels. Provide these patches as input to our trained CNN. Separate out the motion and defocus components from the composite kernel predicted at each pixel by the network.
Use the defocus component to identify the depth layer that has the maximum area and label this as the reference depth layer $d_0$.
Segment the moving objects in $d_0$ from the background region using the blur compatibility criterion.
Estimate $\omega_0$ using a few motion PSFs from the background region in $d_0$.
for depth layers $d_p, p \neq 0$ do
    Segment the moving objects in $d_p$ by checking for consistency between the PSFs estimated using $\omega_0$ and the motion kernels predicted by our network.
end for
Refine the borders of the moving objects using alpha matting.
```

4 Experiments

We evaluate our algorithm's performance on synthetic and real data. For the first set of experiments in this section, we create from the light field saliency dataset [19], our own quasi-real database of dynamic 3D scenes as observed by an out-of-focus moving camera. Next, we study our algorithm's effectiveness in detecting and segmenting dynamic objects using the publicly available blur perception dataset [23]. Finally, we demonstrate our technique's effectiveness on real images that we either downloaded from the internet or captured ourselves using a hand-held camera. We compare our method's performance against two state-of-the-art algorithms [4, 21] for motion segmentation, and provide quantitative and qualitative evaluations for segmentation accuracy. The approach in [4] also detects moving objects, and is the closest competing method. We also show comparisons with the motion segmentation algorithm in [21] although their method only segments the image into different motion segments, and cannot detect whether an object is moving. It is to be noted that the methods in [4, 21] do not account for defocus blur. While the work of [4] is publicly available, the authors of [21] made their code available to us upon our request.

4.1 Quasi-real light field dataset

The light field saliency dataset (LFSD) [19] contains 100 light fields of indoor and outdoor scenes captured using a Lytro light field camera. For each light field, a focal stack of the 3D scene, ground truth pixel-wise annotation of the salient object(s), and the depth map are provided. We selected two images from the focal stack corresponding to each light field so as to obtain a total of 200 images. The two images were chosen such that the first has the foreground in focus while

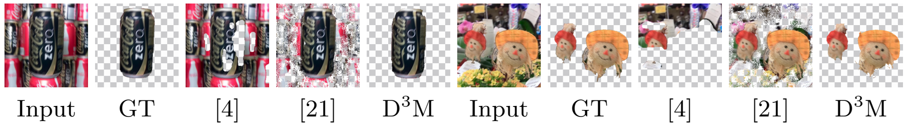

Fig. 5. Segmentation results on two examples from our quasi-real LFSD dataset. GT = ground truth.

the second has the background in focus i.e., the defocus blur varies monotonically from foreground to background. However, there is no motion blur in LFSD since the scene and the camera are static. To bring about the interplay of camera motion, object motion and defocus blur, and to put D^{3}M to the full test, we blur the images synthetically. To mimic real camera shake, we defined the permissible range of camera translations and rotations based on the camera trajectories in [17]. Next, we generated random trajectories for both camera and foreground objects, and created synthetically blurred observations using our composite blur model with the help of the ground truth masks of the foreground salient objects and the depth maps. We selected this dataset in particular for this experiment because the images already contain defocus blur, while the availability of the ground truth mask of the foreground objects and the depth map of the scene allowed us to synthetically add camera and object motion blur. We call this new database 'LFSD quasi-real'. A few representative examples are given in Fig. 5. Our segmentation results post-matting, and the outputs of the methods in [4] and [21] are also shown. For visualization, only the moving objects have been displayed both for the ground truth and the results, while the remaining portions have been masked out by a checkerboad pattern. In the first example, the method in [4] wrongly labels a lot of the background as moving, while it fails to detect one of the dynamic foreground objects in the second. Since the technique in [21] does not classify whether an object is moving or not, but merely partitions the image into various motion segments, we treat the largest segment in their output as the static background, and the remaining segments as belonging to dynamic objects. It can be seen that in both examples, their algorithm picks up a lot of spurious regions in the background that do not belong to the dynamic objects. On the other hand, a comparison with the ground truth reveals that D^{3}M has correctly segmented out the moving objects. Since quantification of segmentation accuracy is very important, we computed precision and recall values averaged

Table 1. Average precision and recall values. For computing precision and recall, 'positive' is when a pixel is classified as being dynamic.

Methods	[4]		[21]		D^{3}M	
Datasets	Precision	Recall	Precision	Recall	Precision	Recall
LFSD quasi-real	0.453	0.335	0.554	0.493	0.856	0.782
[23]	0.367	0.291	0.486	0.419	0.778	0.701

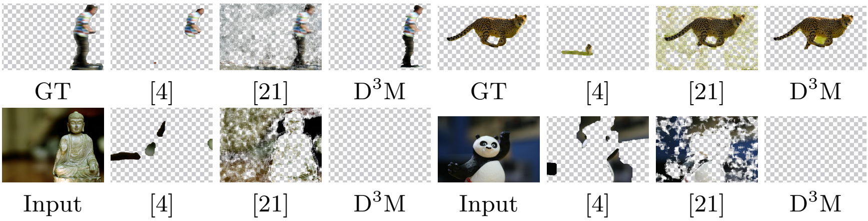

Fig. 6. Segmentation results for the two dynamic examples in Figs. 1(a) and (b) from the blur perception dataset [23] are shown in row one. Row two shows two examples of static scenes from the same dataset.

over all 200 images and these are provided in Table 1. It can be observed that we outperform competing methods by a significant margin.

4.2 Blur perception dataset [23]

The blur perception dataset [23] contains 1000 images with blur due to either camera or object motion, and defocus. Human labeled masks of the blurred regions are available, and the dataset is originally designed for the benchmarking of algorithms that detect blurred pixels in an image. Since our main goal is a quantitative evaluation of our algorithm's dynamic segmentation capabilities, and the database has both static and dynamic scenes, we select for our study only those images (296 in total) which have dynamic object(s). The input images for the two examples in row one of Fig. 6 were already shown in Figs. 1(a) and (b). Fig. 1(a) has a dynamic foreground object imaged by a static camera, while Fig. 1(b) was captured by a panning camera such that the foreground object is not blurred i.e., the relative motion between the camera and the foreground object is zero. In both cases, our algorithm correctly identifies the foreground object which occupies a smaller region than the background as the moving object. We again report precision and recall values averaged over the 296 dynamic images from the dataset of [23] in Table 1. We additionally show two static examples from the same dataset in row two of Fig. 6, and it can be observed that our algorithm unerringly flags all pixels as static.

4.3 Additional examples

In this section, we provide results for the examples in Fig. 3(a), Figs. 1(c) and (d). The first three columns of Fig. 7 show outputs on the synthetic example in Fig. 3(a). The method in [4] wrongly classifies the background region as the moving object while the aircraft and the birds have been labeled as stationary. [21] incorrectly labels even the bird in the middle layer as dynamic because of depth-motion ambiguity. D^{3}M correctly identifies the two aircraft in the background and the bird in the foreground as dynamic.

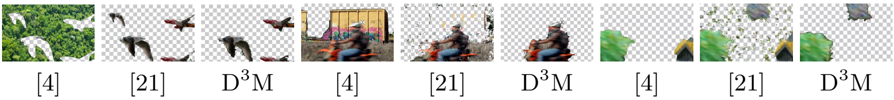

Fig. 7. Segmentation results for the examples in Fig. 3(a), Figs. 1(c) and (d).

The street-side scene in Fig. 1(c) was extracted from a video downloaded from the internet. In this frame, we have observed (based on the video) that the person on the motorbike is moving and is slightly out-of-focus. The results for this example are shown in columns four to six of Fig. 7. D^3M yet again correctly identifies the moving object, while the output of the methods in [4] and [21] erroneously classify a lot of the background region as dynamic.

The results for the indoor image in Fig. 1(d) are displayed in the last three columns of Fig. 7. We captured this image using a hand-held Canon EOS 60D camera. The objects and the background were kept within a distance of two meters from the camera. In Fig. 1(d), there is a moving object at the top center which is at the same depth as the background. The two objects in the foreground (bottom left and bottom right) are out-of-focus. While the object on the bottom left was moving, the object on the bottom right was stationary with respect to the background. Note that the entire scene is blurred due to the motion of the camera. It can be seen from our results that D^3M is not only able to correctly detect the moving objects but also accurately delineate the boundaries. The object on the bottom right is wrongly marked as moving by the method in [4]. Moreover, the moving object at the top center has been incorrectly classified as stationary. The output of [21] falsely detects the object on the bottom right as dynamic. More examples are included in the supplementary material.

5 Conclusions

We proposed a method D^3M to detect moving objects from a single image of a 3D scene affected by camera shake by jointly considering the effects of optical defocus and motion blur. The composite kernel at each pixel was inferred using a CNN. By decoupling the motion and defocus kernels, we can unambiguously segment the moving objects. We validated our proposed framework on public datasets, frames extracted from videos downloaded from the internet, as well as images we captured ourselves using a hand-held camera. The ability of our algorithm to segment and accurately recover the boundaries of dynamic objects was adequately demonstrated. As future work, it would be interesting to separate out the object's independent motion from the combined motion blur kernel which also includes the depth-dependent camera motion component.

References

1. Aizenberg, I., Paliy, D., Zurada, J., Astola, J.: Blur identification by multilayer neural network based on multivalued neurons. IEEE Trans. Neural Networks 19(5), 883–898 (2008)
2. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. 26(9), 1124–1137 (2004)
3. Caglioti, V., Giusti, A.: On the apparent transparency of a motion blurred object. International Journal of Computer Vision 86(2-3), 243–255 (2010)
4. Chakrabarti, A., Zickler, T., Freeman, W.T.: Analyzing spatially-varying blur. In: Proc. CVPR. pp. 2512–2519 (2010)
5. Chaudhuri, S., Rajagopalan, A.N.: Depth from defocus - a real aperture imaging approach. Springer (1999)
6. Dai, S., Wu, Y.: Removing partial blur in a single image. In: Proc. CVPR. pp. 2544–2551 (2009)
7. Deng, X., Shen, Y., Song, M., Tao, D., Bu, J., Chen, C.: Video-based non-uniform object motion blur estimation and deblurring. Neurocomput. 86, 170–178 (2012)
8. Elgammal, A., Duraiswami, R., Harwood, D., Davis, L.: Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. Proceedings of the IEEE 90(7), 1151–1163 (2002)
9. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html
10. Favaro, P., Mennucci, A., Soatto, S.: Observing shape from defocused images. International Journal of Computer Vision 52(1), 25–43 (2003)
11. Favaro, P., Soatto, S.: A variational approach to scene reconstruction and image segmentation from motion-blur cues. In: Proc. CVPR. pp. 631–637 (2004)
12. Favaro, P., Soatto, S.: 3-D Shape Estimation and Image Restoration: Exploiting Defocus and Motion-Blur. Springer-Verlag New York, Inc. (2006)
13. Gupta, A., Joshi, N., Zitnick, C.L., Cohen, M., Curless, B.: Single image deblurring using motion density functions. In: Proc. ECCV. pp. 171–184 (2010)
14. Hu, Z., Yang, M.H.: Fast non-uniform deblurring using constrained camera pose subspace. In: Proc. BMVC. pp. 1–11 (2012)
15. Hu, Z., Yang, M.H.: Good regions to deblur. In: Proc. ECCV. pp. 59–72 (2012)
16. Jia, J.: Single image motion deblurring using transparency. In: Proc. CVPR. pp. 1–8 (2007)
17. Köhler, R., Hirsch, M., Mohler, B., Schölkopf, B., Harmeling, S.: Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In: Proc. ECCV. pp. 27–40 (2012)
18. Levin, A., Lischinski, D., Weiss, Y.: A closed-form solution to natural image matting. IEEE Trans. Pattern Anal. Mach. Intell. 30(2), 228–242 (2008)
19. Li, N., Ye, J., Ji, Y., Ling, H., Yu, J.: Saliency detection on light field. In: Proc. CVPR. pp. 2806–2813 (2014)
20. Paramanand, C., Rajagopalan, A.: Depth from motion and optical blur with an unscented Kalman filter. IEEE Trans. Image Processing 21(5), 2798–2811 (2012)
21. Paramanand, C., Rajagopalan, A.: Motion blur for motion segmentation. In: Proc. ICIP. pp. 4244–4248 (2013)
22. Sheikh, Y., Shah, M.: Bayesian modeling of dynamic scenes for object detection. IEEE Trans. Pattern Anal. Mach. Intell. 27(11), 1778–1792 (2005)

23. Shi, J., Xu, L., Jia, J.: Discriminative blur detection features. In: Proc. CVPR. pp. 2965–2972 (2014)
24. Shi, J., Xu, L., Jia, J.: Just noticeable defocus blur detection and estimation. In: Proc. CVPR. pp. 657–665 (2015)
25. Shroff, N., Veeraraghavan, A., Taguchi, Y., Tuzel, O., Agrawal, A., Chellappa, R.: Variable focus video: Reconstructing depth and video for dynamic scenes. In: Proc. ICCP. pp. 1–9 (2012)
26. Sorel, M., Flusser, J.: Space-variant restoration of images degraded by camera motion blur. IEEE Trans. Image Processing 17(2), 105–116 (2008)
27. Sun, J., Cao, W., Xu, Z., Ponce, J.: Learning a convolutional neural network for non-uniform motion blur removal. In: Proc. CVPR. pp. 769–777 (2015)
28. Tai, Y.W., Tan, P., Brown, M.S.: Richardson-Lucy deblurring for scenes under a projective motion path. IEEE Trans. Pattern Anal. Mach. Intell. 33(8), 1603–1618 (2011)
29. Vedaldi, A., Lenc, K.: Matconvnet – convolutional neural networks for matlab (2015)
30. Whyte, O., Sivic, J., Zisserman, A., Ponce, J.: Non-uniform deblurring for shaken images. International Journal of Computer Vision 98(2), 168–186 (2012)
31. Yan, R., Shao, L.: Image blur classification and parameter identification using two-stage deep belief networks. In: Proc. BMVC (2013)
32. Zhuo, S., Sim, T.: Defocus map estimation from a single image. Pattern Recogn. 44(9), 1852–1858 (2011)