ESM-Blur: Handling & Rendering Blur in 3D Tracking and Augmentation

Youngmin Park* GIST, U-VR Lab
Vincent Lepetit† EPFL, CVLab
Woontack Woo‡ GIST, U-VR Lab

Abstract

The contribution of this paper is two-fold. First, we show how to extend the ESM algorithm to handle motion blur in 3D object tracking. ESM is a powerful algorithm for template matching-based tracking, but it can fail under motion blur. We introduce an image formation model that explicitly considers the possibility of blur, and show it results in a generalization of the original ESM algorithm. This allows to converge faster, more accurately and more robustly even under large amount of blur. Our second contribution is an efficient method for rendering the virtual objects under the estimated motion blur. It renders two images of the object under 3D perspective, and warps them to create many intermediate images. By fusing these images we obtain a final image for the virtual objects blurred consistently with the captured image. Because warping is much faster that 3D rendering, we can create realistically blurred images at a very low computational cost.

1 Introduction

With the fast increasing popularity of webcam- and mobile devices-based AR applications, motion blur is one of the main obstacles for vision-based tracking and augmentation. Because it makes salient features almost disappear, motion blur disturbs tracking algorithms based on corners or edges extraction. One solution is to use detection algorithms [7, 11] to restart tracking when the blur stops, but the interruption spoils the user experience.

A method that explicitly handles motion blur is therefore desirable. For example, [10] handles blur in marker-based tracking, and [6] in natural features-based camera tracking. Camera tracking can exploit the whole image, while we mainly focus here on object tracking as, which is more challenging: It can only rely on the image part containing the object, and in presence of blur, the amount of image information relevant for tracking can become very low. Like us, [4, 9] consider motion blur in the case of template tracking. The main advantages of our approach is its simplicity and efficiency. We show how to easily modify the ESM algorithm, the most efficient template matching algorithm [2], without any need for blurring the template.

Moreover, for Augmented Reality applications, another problem occurs since the virtual objects must be rendered to reflect correctly the motion blur, otherwise they would look unnatural. [3, 5] are limited to 2D translational motions. [9] proposes an algorithm that take perspective into account correctly and shows it can efficiently be implemented on the GPU, unfortunately it can only consider planar objects. While simple, our method efficiently synthesizes motion blur on general 3D objects.

As Figure 1 shows, our contribution is two-fold. We propose a method for object tracking even in presence of large amounts of motion blur, and also a method for rendering blurred virtual objects under general 3D motion. For tracking we explicitly introduce motion blur into the image formation model for template matching.

*e-mail: ypark@gist.ac.kr
†e-mail: vincent.lepetit@epfl.ch
‡e-mail: wwoo@gist.ac.kr

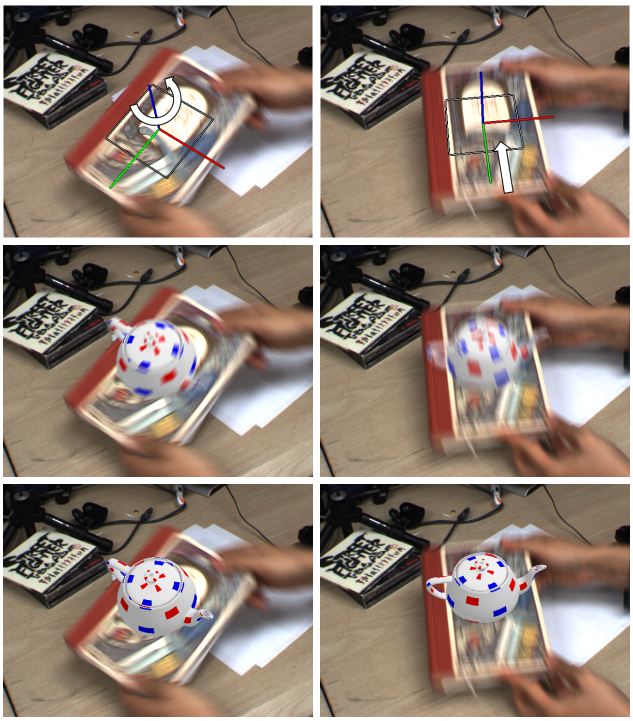


Figure 1: Tracking and motion blur generation for various 3D motions for (left:) a 3D rotation, (right:) a 3D translation. **Top row:** Our algorithm can track the book despite heavy motion blur. The arrows depict the dominant motion in each figure. **Middle row:** A virtual teapot is rendered with consistent motion blur. Our method properly generates motion blur for general motion, including 3D rotations. **Bottom row:** For comparison, the same teapot rendered without motion blur.

For rendering we rely on the same image formation model. Our algorithm avoids extensive repetitive rendering required to generate motion blur for 3D motion by combining 2D warping and 3D rendering.

We experimented with several real sequences in which the tracked object moves fast and presents heavy motion blur. We obtain faster, more robust, and more accurate convergence compared to the standard ESM algorithm. We also show the generated motion blur reflects successfully the 3D motion.

In the remainder of the paper, we first describe our methods for motion blurred object tracking and motion blur generation, and present our results.

2 The ESM-Blur Algorithm

In this section, we first introduce our image formation model that considers blur explicitely, and we then show how the ESM algorithm can be adapted to this model. In the next section we will show how to efficiently render blurred virtual objects under the same image formation model.

2.1 Image Formation Model

The usual approach of template matching-based tracking techniques is to assume a simple image formation model $\mathcal{I}(\Theta)$ where

the captured image $\mathcal{I}_c$ can be generated by warping a known template $\mathcal{T}$:

$$\mathcal{I}(\Theta) = w(\mathcal{T}, \Theta)\,, \tag{1}$$

where $w(.,.)$ is a warping function, here applied to the template $\mathcal{T}$ with parameters Θ. The goal is to estimate the motion $\widehat{\Theta}$ by maximizing the correlation between the captured image $\mathcal{I}_c$ and the warped template as:

$$\widehat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \|\mathcal{I}_c - \mathcal{I}(\Theta)\|^2\,. \tag{2}$$

However, this is valid only when the captured image is not blurred. To correctly handle blur, we generalize the image formation model by writing:

$$\mathcal{I}(m) = \frac{1}{t_1 - t_0}\int_{t_0}^{t_1} w(\mathcal{T}, m(t))dt\,, \tag{3}$$

that is, the captured image is made of the average of the template appearances when it is warped under a motion defined by $m(t)$. t_0 denotes the time when the camera shutter opens, and t_1 the time when it closes. Instead of estimating a single motion vector Θ, we now have to estimate the full motion $m(t)$ of the template between times t_0 and t_1. To make the estimation possible, we will assume that this motion is linear so that we can write:

$$m(t) = \frac{t}{t_1}\Theta\,, \tag{4}$$

where Θ is the template motion at time t_1, and our image formation model of Eq. (3) simplifies in:

$$\mathcal{I}(\Theta) = \frac{1}{t_1 - t_0}\int_{t_0}^{t_1} w(\mathcal{T}, \frac{t}{t_1}\Theta)dt\,. \tag{5}$$

This model is valid even if there is no motion blur—in cases when neither the object nor the camera move. Moreover when $t_0 \to t_1$, that is when the capture is assumed to be instantaneous, this model tends to the more standard model of Eq. (1). We show in the following that optimizing Eq. (2) can be done for such a model.

2.2 Optimization

As shown in [1], we can always assume that the template motion Θ is small. As in the original ESM algorithm, we first consider the second-order Taylor expansion of the warping function. To make the derivations simpler, we consider here the simple case where $\Theta = \mathbf{0}$ corresponds to the template reference position. However, the derivations can be extended to the more general case as it was done in [8] in a straightforward way. We can therefore write:

$$w(\mathcal{T}, \Theta) \approx \mathcal{T} + \mathbf{J}_{\mathcal{T}}\Theta + \Theta^\top \mathbf{H}_{\mathcal{T}}\Theta\,, \tag{6}$$

where $\mathbf{J}_{\mathcal{T}}$ is the Jacobian of $w(\mathcal{T},.)$ computed at $\Theta = \mathbf{0}$, and $\mathbf{H}_{\mathcal{T}}$ is its Hessian, also computed at $\Theta = \mathbf{0}$. $\mathbf{H}_{\mathcal{T}}$ is of rather unusual nature since it is a tensor, but fortunately we won't have to estimate it. By plugging Eq. (6) into Eq. (5), we obtain the following second-order approximation of our image formation model:

$$\begin{aligned}
& \mathcal{I}(\Theta) \\
\approx\ & \frac{1}{t_1 - t_0}\int_{t_0}^{t_1} [\mathcal{T} + \mathbf{J}_{\mathcal{T}}\left(\frac{t}{t_1}\Theta\right) + \left(\frac{t}{t_1}\Theta\right)^\top \mathbf{H}_{\mathcal{T}}\left(\frac{t}{t_1}\Theta\right)]dt \\
=\ & \mathcal{T} + \frac{1}{t_1(t_1 - t_0)}(\int_{t_0}^{t_1} t dt)\mathbf{J}_{\mathcal{T}}\Theta + \frac{1}{t_1^2(t_1 - t_0)}(\int_{t_0}^{t_1} t^2 dt)\Theta^\top \mathbf{H}_{\mathcal{T}}\Theta \\
=\ & \mathcal{T} + a\mathbf{J}_{\mathcal{T}}\Theta + b\Theta^\top \mathbf{H}_{\mathcal{T}}\Theta
\end{aligned} \tag{7}$$

where a and b depend on t_0 and t_1.

ESM can make the second-order term $\Theta^\top \mathbf{H}_{\mathcal{T}}$ disappear by considering the expression of the Jacobian of $w(\mathcal{T},.)$ for the captured image. In our case, it is more difficult since this image is blurred. However, we can still express the Jacobian $\mathbf{J}_{\mathcal{I}}$ of our image formation function $\mathcal{I}(\Theta)$ at Θ by computing the derivative of Eq. (7). We get:

$$\mathbf{J}_{\mathcal{I}}(\Theta) = a\mathbf{J}_{\mathcal{T}} + 2b\Theta^\top \mathbf{H}_{\mathcal{T}}\,. \tag{8}$$

By plugging Eq. (8) into Eq. (7), we can avoid the awkward term $b\Theta^\top \mathbf{H}_{\mathcal{T}}$ in the expression of our image formation model:

$$\begin{aligned}
\mathcal{I}(\Theta) &\approx \mathcal{T} + a\mathbf{J}_{\mathcal{T}}\Theta + \frac{1}{2}(\mathbf{J}_{\mathcal{I}}(\Theta) - a\mathbf{J}_{\mathcal{T}})\Theta \\
&= \mathcal{T} + \frac{1}{2}(\mathbf{J}_{\mathcal{I}}(\Theta) + a\mathbf{J}_{\mathcal{T}})\Theta\,,
\end{aligned}$$

which we can use to compute the motion $\widehat{\Theta}$ as:

$$\widehat{\Theta} = \mathbf{J}^+_{\text{blur}}(\mathcal{I}_c - \mathcal{T})\,.$$

$\mathbf{J}^+_{\text{blur}}$ is the pseudo-inverse of $\mathbf{J}_{\text{blur}}$, with

$$\mathbf{J}_{\text{blur}} = \frac{1}{2}(\mathbf{J}_{\mathcal{I}}(\Theta) + a\mathbf{J}_{\mathcal{T}}) \tag{9}$$

and

$$a = \frac{1}{t_1(t_1 - t_0)}(\int_{t_0}^{t_1} t dt) = \frac{t_0 + t_1}{2t_1}\,. \tag{10}$$

Eq. (9) generalizes the standard ESM formula. When we assume as it is usually done that the exposure time is infinitesimal, then $t_0 = t_1$ and $a = 1$, and Eq. (9) simplifies to the standard ESM formula. On the other extreme, when the shutter remains opened, $t_0 = 0$ and then $a = \frac{1}{2}$, reducing the influence of the Jacobian computed at the template.

It is therefore very easy to modify an existing implementation of the original ESM to make it more robust to motion blur. In practice, the correct value for a can be directly computed if one knows the camera shutter opening and closing times. For the results presented in this paper the camera was set so that the camera shutter remains open, and we took $a = \frac{1}{2}$.

2.3 Initialization and Re-Initialization

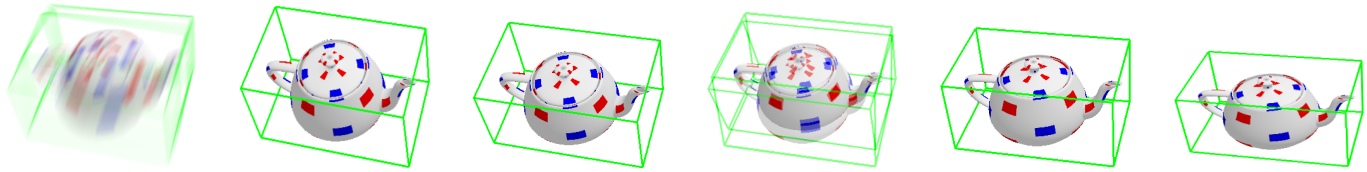
We use the Ferns method [11] to automatically initialize our tracking method described above. It also happens that tracking fails when the blur is too important or when there is a complete occlusion, and we use the Ferns for re-initialization when a failure is detected. To detect a failure, we compute the Normalized Cross-Correlation (NCC) between the template and the image patch warped back from the input image with the estimated pose. In practice, we estimate the tracking has failed if the NCC is less than 0.8.

3 3D Motion-Blur Generation

This section describes our method to render the virtual objects under a motion blur consistent with the blur present in the captured image. It is based on the image formation model already introduced in Eq. (3) for tracking. To stick with real-time constraints, we propose a method that combines 3D rendering and 2D warping for efficiency.

3.1 Rendering Blur

One general way to simulate motion blur is to render the virtual object many times along its path during the image exposure and blend the rendered images all together. However, several tens of such images are needed in practice to create realistic smooth blur, and doing the rendering in 3D would be prohibitive for real-time applications. For example, rendering a *single* image of the GLUT



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

Figure 2: Overview of our approach to efficient blur rendering. To render the virtual object consistently with the retrieved motion such as in (a), we first generate two intermediate views (c) and (e) we call *intraframes* using OpenGL. To create the other intermediate views, we warp the intraframes using affine transformations to approximate perspective projections. The contributions of the two intraframes are weighted and summed to obtain intermediate views such as (d). (b) and (f) are other intermediate views for the pose that are not in-between the two intraframes. Finally, by summing all the intermediate views, we obtain a blurred view as in (a).

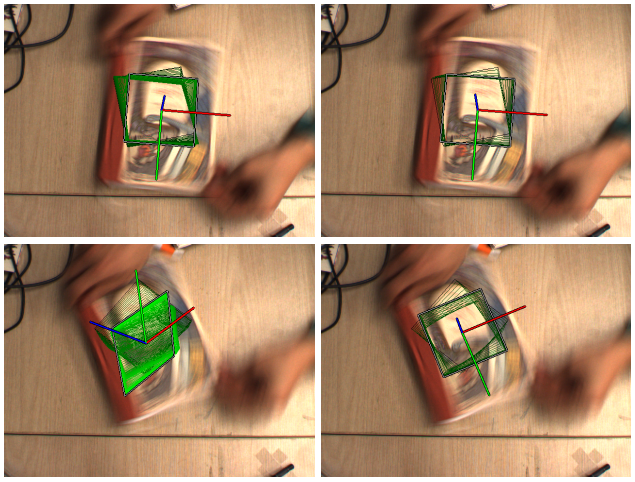


Figure 3: Comparison of the convergences of standard ESM (left column) and our method (right column) on difficult images. The quadrangles denote the intermediate positions for the template as retrieved by the successive iterations. **Top row:** Both methods estimate the correct pose, however our method converges much faster. **Bottom row:** Our method converges while standard ESM diverges, usually after many iterations.

teapot using the `glutSolidTeapot` function takes about 3 ms on our desktop PC. Rendering more complex objects in practical applications would take longer.

Therefore, as depicted by Figure 2, our strategy is to render only a few images of the virtual object, and generate intermediate images by warping these rendered images. We call such sparsely rendered images *intraframes* by quoting the term from the MPEG format specifications. Because warping is much faster than 3D rendering, this allows us to generate correctly blurred images within our real-time constraints.

More formally, the image $\mathcal{I}_{\text{virt}}$ we want to create can be written as:

$$\mathcal{I}_{\text{virt}} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} r(\mathcal{O}, m(t)) dt \,, \tag{11}$$

where $r(\mathcal{O}, m(t))$ is the image of the virtual object $\mathcal{O}$ rendered under pose $m(t)$. As in Section 2, t_0 denotes the time when the camera shutter opens, t_1 the time when it closes. Also as in Section 2, we assume that $m(t)$ evolves linearly between t_0 and t_1. Our image formation model for rendering is therefore similar to the one we used for tracking as defined in Eq. (3). But contrary to Section 2, we cannot assume here the object pose is always small, and we write $m(t)$ as $m(t) = \Theta_p + \frac{t}{t_1}\Theta$, where Θ_p is the object pose estimated for the previous captured image, and Θ is the object motion for the current image computed as in the previous section.

In practice, we have to replace the integral in Eq. (11) by a finite sum:

$$\mathcal{I}_{\text{virt}} \approx \frac{1}{n+1} \sum_{i=0}^{n} r(\mathcal{O}, \Theta_p + \frac{i}{n}\Theta) \,. \tag{12}$$

The blur realism increases with n, but of course the rendering time increases as well. In practice we use $n = 50$. As discussed above, we actually render in 3D only two intraframes. For a good balance, we choose to take them at $i = \frac{n}{4}$ and $i = \frac{3n}{4}$. We therefore approximate $\mathcal{I}_{\text{virt}}$ as

$$\mathcal{I}_{\text{virt}} \approx \frac{1}{n+1} (\mathbf{I}_{\frac{n}{4}} + \mathbf{I}_{\frac{3n}{4}} + \sum_{\substack{i=0 \\ i \neq \frac{n}{4}, i \neq \frac{3n}{4}}}^{n} \mathbf{B}_i) \,, \tag{13}$$

where $\mathbf{I}_{\frac{n}{4}} = r(\mathcal{O}, \Theta_p + \frac{\Theta}{4})$ and $\mathbf{I}_{\frac{3n}{4}} = r(\mathcal{O}, \Theta_p + \frac{3\Theta}{4})$. The images $\mathbf{B}_i$ for the other values of i are intermediate images created from $\mathbf{I}_{\frac{n}{4}}$ and $\mathbf{I}_{\frac{3n}{4}}$ as explained in the next section.

3.2 Generating the Intermediate Images

From the two intraframes, we generate n intermediate frames using affine warping. One strategy could be to use only the first intraframe to generate the first part of the intermediate frames, and the second intraframe to generate the second part. However that would result in a discontinuity at the junction of the two parts since the affine warping is only an approximation of a real 3D rendering. Near this junction, we therefore blend the warped images from the two intraframes together to avoid this discontinuity.

More exactly an intermediate frame $\mathbf{B}_i$ is computed as

$$\mathbf{B}_i = \begin{cases} w(\mathbf{I}_{\frac{n}{4}}, \mathbf{A}_{\frac{n}{4} \to i}) & \text{if } i < \frac{n}{4} \\ \alpha(i) w(\mathbf{I}_{\frac{n}{4}}, \mathbf{A}_{\frac{n}{4} \to i}) + \beta(i) w(\mathbf{I}_{\frac{3n}{4}}, \mathbf{A}_{\frac{3n}{4} \to i}) & \text{if } \frac{n}{4} < i < \frac{n}{4} \\ w(\mathbf{I}_{\frac{3n}{4}}, \mathbf{A}_{\frac{3n}{4} \to i}) & \text{if } i > \frac{3n}{4} \end{cases} \tag{14}$$

where $\alpha(i)$ varies linearly from 1 to 0 when i varies from $\frac{n}{4}$ to $\frac{3n}{4}$, and $\beta(i)$ varies linearly from 0 to 1 when i varies from $\frac{n}{4}$ to $\frac{3n}{4}$. $w(.,.)$ is the warping function and the $\mathbf{A}_{i \to j}$ are the affine transformations between image i and image j.

To compute the affine transformations $\mathbf{A}_{i \to j}$, we rely on the projections of the corners of the bounding box to the virtual object. We project these corners in the intraframes and the intermediate frames using a pose interpolated in 3D from t_0 to t_1. The transformation $\mathbf{A}_{i \to j}$ is then computed as the affine transformation that transforms the corners projections in frame i to their projections in frame j. Thus the transformation is approximately valid for any pixel lying on the virtual object.

3.3 Implementation

For efficiency, we warp the intraframes and blend the resulting intermediate images using the graphic card only. Not only the graphic card can perform these two operations very fast, but also this avoids

Table 1: Average times for tracking and rendering. The standard ESM algorithm took 5.62 ms in average for motion estimation. Rendering was performed using 48 intermediate images, and the naive solution with 50 views takes about 171.75 ms.

	Step	Average time (ms)
Tracking	Conversion to grayscale	0.89
	Motion estimation	0.18 / iter (4.20 / frame)
Rendering	Intraframe rendering	7.27
	Inter. image & alpha blending	0.70
	Final rendering	1.00

reading back the intraframes from the video memory into the main memory. The rendering of the intraframes is done using render-to-texture in OpenGL. The textures are set to fit on a virtual image plane, which is then transformed by linear interpolation to create the intermediate images. The final motion-blurred image is generated by simply accumulating each image with proper alpha values. The whole process is very fast and enough images can be generated to create a realistic blur effect while keeping the real-time constraint.

4 Experimental Results

We experimented our tracking and motion blur generation algorithm with a desktop PC having 3.2 GHz CPU and NVIDIA Quadro FX 5600 graphics card. Figure 1 shows the overall result of our tracking and rendering methods. The video is captured at 30 Hz with the shutter speed of 33.33ms so as to the shutter is almost always opened while grabbing a frame. Therefore, we set $t_0 = 0$ and $t_1 = 33.33$ms, giving a value of $\frac{1}{2}$ for a in Eq. (10).

The tracked template is 10×10 cm large, and taken at the center of the book cover. We moved the book freely including rotation and translation along three axes. The tracking method estimated the motion successfully even though the book is heavily motion-blurred due to fast movement and long shutter speed. For the various motions, the teapot is correctly motion-blurred.

Our test sequence is 752 frames long, and is available as a supplementary material. Some frames are presented in Figure 1. Over this sequence, ESM fails to converge correctly 11 times, while ESM-Blur fails only 5 times. Both methods are reinitialized automatically in case of failure using a Fern-based detector when the amount of blur becomes small enough, however avoiding failures is of course still desirable to avoid augmentation interruption.

Figure 4 compares our ESM-Blur method and standard ESM over a section of the sequence shown Figure 1 for which both methods always converge correctly. The iterations were stopped when the Euclidean norm of the motion update was less than 0.01. The frames with more motion blur typically need more iterations and yield lower NCC. ESM-Blur always converges faster, with an average number of iterations of 31.21 for ESM and 23.31 for ESM-Blur. In absence of ground truth data, we evaluated the accuracy with the Normalized Cross-Correlation between the template patch and the patch in the input image. Our algorithm almost always yields higher correlation. Computation times for the most consuming steps are given in Table 1. Because of fewer iterations, ESM-Blur is faster.

Figure 1 shows augmented images with and without rendering motion blur. Motion-blur clearly improves the integration realism. The most time consuming and object dependent part is the intraframe rendering, which needs to draw the object twice. By contrast, rendering intermediate frames is very cheap and independent from the complexity of the object. Both operations are performed on the graphics card, letting the CPU acquiring and tracking the next frame while the GPU is rendering the augmented image.

5 Conclusion & Future work

We proposed two complementary methods for Augmented Reality under motion-blur, one for tracking, and the other one for rendering. Our approach is general: We first introduced a suitable image formation model, to easily extend an existing efficient algorithm. It would be interesting to investigate other image formation models to better handle complex imaging effects, for example image distortions due to rolling shutters often used in webcams and cameras in mobile phones.

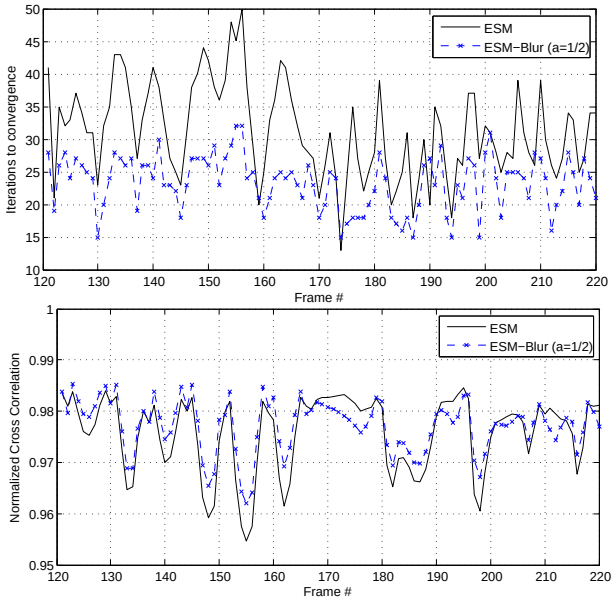


Figure 4: Comparison between ESM and our method (ESM-Blur) over a section of the sequence shown Figure 1 for which both methods always converge. **Top:** The number of iterations needed for both methods. Our method converges within less iterations. **Bottom:** The normalized cross correlation between the warped template patch and the current image. Our algorithm almost always yields higher correlation.

Acknowledgements

This research was supported in part by the Practical development project of GIST GTI and in part by the CTI development project of KOCCA, MCST in S.Korea.

References

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, Mar. 2004.

[2] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *IJRR*, 2007.

[3] J. Fischer, D. Bartz, and W. Strasser. Enhanced visual realism by incorporating camera image effects. In *ISMAR'06*.

[4] H. Jin, P. Favaro, and R. Cipolla. Visual tracking in the presence of motion blur. In *CVPR'05*.

[5] G. Klein and D. Murray. Compositing for small cameras. In *ISMAR'08*.

[6] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *ECCV'08*.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, Nov. 2004.

[8] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *ICRA'04*.

[9] C. Mei and I. Reid. Modeling and generating complex motion blur for real-time tracking. In *CVPR'08*.

[10] B. Okumura, M. Kanbara, and N. Yokoya. Augmented reality based on estimation of defocusing and motion blurring from captured images. In *ISMAR'06*.

[11] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *CVPR'07*.