Supplementary Material

Image Registration and Change Detection under Rolling Shutter Motion Blur

In this supplementary material, we show (i) evaluation of camera motion estimation in the absence of changes, (ii) change detection in planar scenes and comparisons, (iii) change detection in global shutter motion blur images, (iv) handling illumination variation, (v) the effect of size of the changes, and (vi) evaluation metrics.

1 Camera Motion Estimation

We first compare the performance of our method with other methods on camera motion estimation (with no changes in the scene). In Fig. 14, we show two images, a reference clean image in Fig. 14(a) and an image of the same scene distorted with RS and MB in Fig. 14(b). The camera motion applied synthetically on the image is in-plane translatory motion but of nonuniform velocity. The scene is assumed to be planar with only one layer.

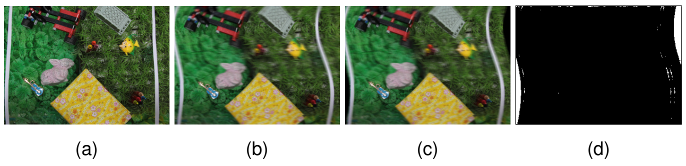


Fig. 14. Synthetic experiment: Camera motion estimation. (a) Reference image, (b) RSMB image, (c) Registered image, and (d) Difference image.

To register these two images, we need to solve (13). We consider the change vector to be absent here to study the efficiency of our camera motion estimation. In general, our method does not require the prior information about the presence or absence of changes in the scene. The change vector values will be zero if there are no changes in the scene. In this experiment, we ignore the identity matrix part of $\mathbf{B}$ and estimate only $\boldsymbol{\omega}$. We start with estimating the camera motion for the middle row assuming the following large pose space $\mathcal{S}^{(M/2)}$: x-translation $N(0, 20, 1)$ pixels, and y-translation $N(0, 20, 1)$ pixels. We restrict the out-of-plane translation values and rotation values to zero. Once the camera pose weight vector is estimated for the middle row, we select the following neighbourhood search space for the other rows following Section 2.4: x-translation $N(t_{cx}, 3, 1)$ pixels, and y-translation $N(t_{cy}, 3, 1)$ pixels. The registered image is got by applying the estimated camera motion on the reference image and is shown in Fig. 14(c). The thresholded difference between the registered image and the distorted image is shown in Fig. 14(d).

Comparisons: We first deblur the RSMB image using [17]; this result is shown in Fig. 15(a). We then estimate the camera motion between the reference and deblurred images using the RS methods of [20] and [22]. We then apply the camera motion on the reference image to register with the deblurred image. The registered images using [20] and [22] are shown in Figs. 15(b) and (c), respectively. The difference images of (a)-(b) and (a)-(c), shown respectively in Figs. 15(d) and (e), contain significant nonzero values depicting misregistration. The RMSEs for our method, [17]+[20], and [17]+[22] in the valid region of the registered images are 2.22, 7.46, and 6.87, respectively.

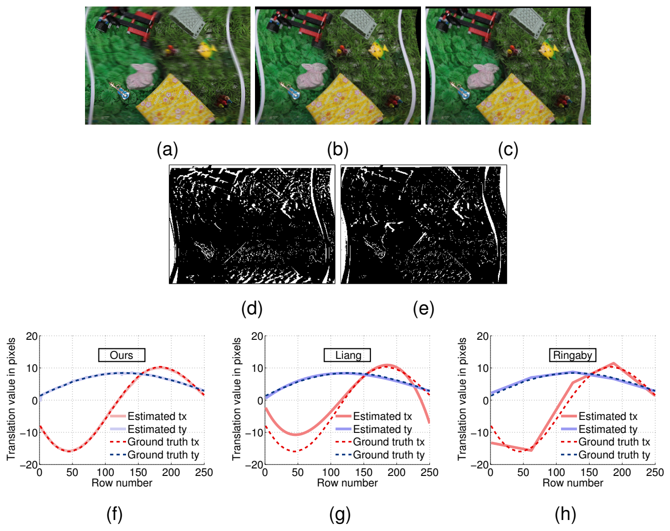


Fig. 15. Comparison with deblur-register MB-RS framework: (a) Deblurred image using [17]. Registered images after RS motion estimation using (b) [20], and (c) [22], and Difference images (d): (a)-(b), (e): (a)-(c). Camera motion estimation for nonuniform camera path using (f) our method, (g) [20], and (h) [22].

In Fig. 15(f-h), we show the camera motion trajectories estimated by our method and other methods. The camera motion estimated by our method, shown in Fig. 15(f) as a continuous line, follows correctly the ground truth trajectory shown as a dotted line. The camera motion estimated by the methods of [20] and [22] are given in Figs. 15(g) and (h), respectively. Since the camera motion has a nonuniform velocity, both the competing methods result in inaccurate motion estimation. Liang's method fits a Bézier curve for the trajectory using the estimated local motion vectors, which results in incorrect fitting. Ringaby's method follows the ground truth trajectory, but it is inaccurate due to the use of interpolation. The RMSEs of the translation motion estimates (t_x,t_y) in pixels of Liang's, Ringaby's, and our method are: (3.45,0.28), (1.79,0.37), and (0.18,0.14), respectively, for the range of [-15,15]pixels for t_x and [0,10]pixels for t_y. The corresponding maximum errors are (8.81,0.81), (5.29,0.82), and (0.80,0.44), respectively. Clearly, the performance of our method is better. Since there are no ground truth changes, the percentage of wrong classification (PWC) provides a quantitative measure for number of estimated wrong changes. The PWC of Liang's, Ringaby's, and our method are 13.9, 10.4, and 0.3, respectively, which indicates that our method detected very less false positives. However, both these methods work well if the camera velocity is uniform *and* the motion blur is negligible.

To confirm the correctness of our implementations of Liang [20] and Ringaby [22], we show an example in Fig. 16 involving registration of an RS-only image. The simulated camera velocity is *uniform*, and each row of the RS image in Fig. 16(b) corresponds only to a single warp of the reference image in Fig. 16(a). The registered image using our method and the other two methods are shown in Figs. 16(c) to (e). The corresponding RMSEs are 0.54, 3.58, 3.67, which show that the registrations using all the three methods are good.

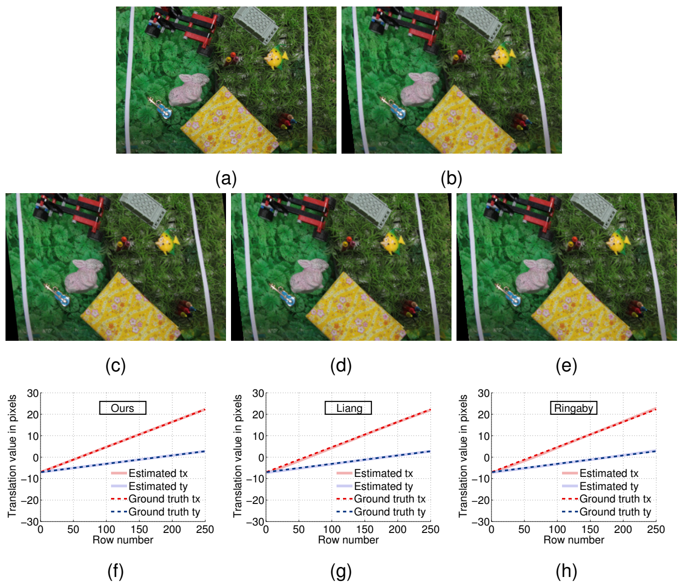


Fig. 16. Synthetic experiment: (a) Reference image, (b) RS image caused by uniform camera motion, and Registered images using (c) our method, (d) [20], and (e) [22]. Camera motion estimation for uniform camera velocity using (f) our method, (g) [20], and (h) [22].

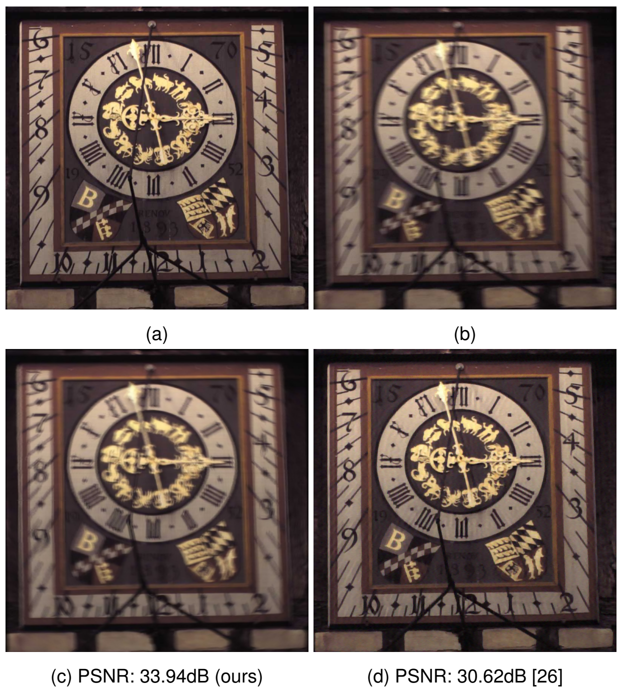


Fig. 17. Synthetic experiment: (a) Reference image, (b) RSMB image caused by 2D out-of-plane rotational motion, (c) registered image using our method, and (d) deblurred image of (a) using [26].

Competing methods perform well in the presence of only RS effect with negligible motion blur provided the camera velocity is uniform. In Figs. 16(f)-(h), we show the estimated camera motions for the simulated camera path. Our method estimates the camera trajectory very well, and so do the other two methods.

We also compare with the joint RSMB framework of [26] for the no-change scenario. We use the images from the synthetic examples provided in [26] for which the ground truth is available. Fig. 17(a) shows the reference image, Fig. 17(a) shows the RSMB image. The RSMB image contains only predominant 2D out-of-plane rotational motion and the in-plane rotational motion is very much negligible. We estimate row-wise motion using (a) and (b) and the RSMB registered image using our algorithm is shown in Fig. 17(c). The PSNR between Figs. 17(b) and (c) is 33.94dB corresponding to an RMSE of 5.12, which shows that the RSMB registration is correct. The deblurred output of Fig. 17(b) using [26] is shown in (d), and the PSNR between the reference image (a) and the deblurred image (d) is 30.62dB, which is also quite high (corresponding to an RMSE of 7.51) indicating correct deblurring. Thus, our method works as good as [26] for the case of 2D motion; for motion having inplane rotations, [26] performs poorly as will be discussed in the next section.

2 Change Detection in Planar Scenes

We now demonstrate our change detection method for planar scenes. We add new objects to the reference image in Fig. 18(a) as changes, and simulate a 3D in-plane camera motion, i.e. in-plane translations (t_x, t_y) and in-plane rotation (r_z). The RSMB image with objects is shown in Fig. 18(b). We solve (13) to estimate both the camera motion weight vector and the change vector. The pose space for the middle row is $N(0, 10, 1)$ pixels for translations, and $N(0, 5, 1)^\circ$ for rotation. The search space for other rows around the estimated

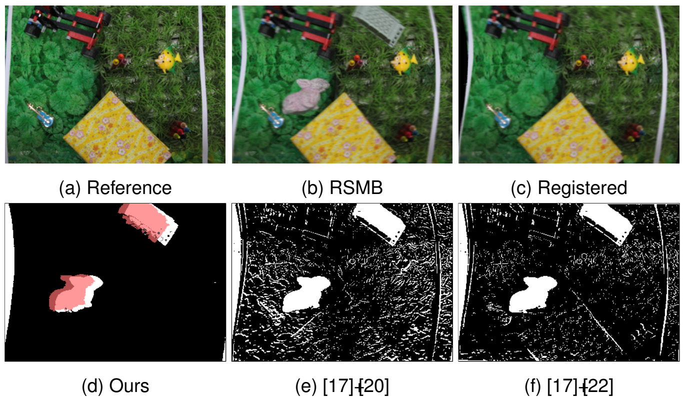


Fig. 18. Synthetic experiment: Change detection in a planar scene. (a) Reference image, (b) RSMB image, (c) Registered image, and (d) Detected changes by our method (marked in white in RSMB grid, and marked in red in reference grid). Detected changes by the method of (e) [17]+[20] and (f) [17]+[22].

centroid of their neighbour rows is $N(t_c, 3, 1)$ pixels for translations, and $N(r_c, 2, 0.5)^\circ$ for rotation. The registered image and detected changes are shown in Figs. 18(c) and (d), respectively. The white regions in Fig. 18(d) show that the detected changes are indeed correct. Note that it is also possible to obtain the locations of changes with respect to the reference image grid by marking pixels in the reference grid which map to the pixels marked as changes in the RSMB grid when the estimated camera motion is applied. The changes thus detected with respect to the reference grid are shown as a red overlay in Fig. 18(d). Figs. 18(e) and (f) show results using the deblur-register frameworks of [17]+[20] and [17]+[22], respectively. There are a high number of false positives in their outputs due to the lack of joint RSMB formulation.

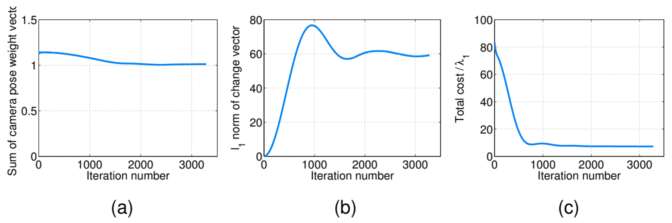


Fig. 19. Various energy values in our cost function in (13) while estimating motion for row number 121 of Fig. 18(b).

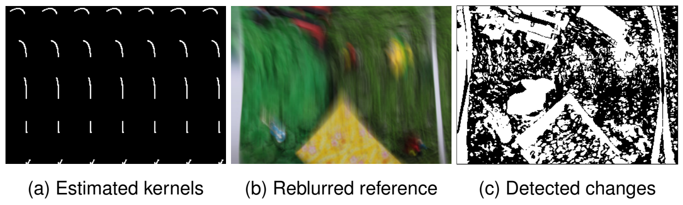


Fig. 20. Comparison with joint RSMB framework [26]: (a) Estimated local blur kernels, (b) reblurred reference image, (c) detected changes.

We plot the various energy values in the cost function while solving (13) for 121st row (randomly chosen) in Fig. 19. Fig. 19(a) shows the value of $\|\boldsymbol{\omega}^{(121)}\|_1$ over iterations. This value converges to one conforming to the conservation of photometric energy. Fig. 19(b) shows the plot of $\|\boldsymbol{\chi}^{(121)}\|_1$ in which the changed pixels are picked correctly over iterations which can be observed by its stabilisation. The plot of the total cost (as a ratio of λ_1) over iterations in Fig. 19(c) shows the convergence of our algorithm.

To compare with the joint RSMB deblurring method [26], we first estimate the camera motion from the RSMB image, and then reblur the reference image using the estimated motion. This is to avoid artifacts due to the deblurring process. Fig. 20(a) shows the estimated local blur kernels, and Fig. 20(b) shows the reblurred image. Since the method does not model the complete camera motion (no inplane rotations), the reblurring result is very different from the RSMB image as can be seen from the detected changes in Fig. 20(c).

2.1 Change Detection – Additional Examples

We capture an image from atop a light house looking down at the road below. The reference image in Fig. 21(a) shows straight painted lines and straight borders of the road. The RSMB image is captured by rotating the mobile phone camera prominently around the y-axis (vertical axis). This renders the straight lines curved as shown in Fig. 21(b). Our algorithm works quite well to register the reference image with the RSMB image as shown in Fig. 21(c). The new objects, both the big vehicles and smaller ones, have been detected correctly as shown in Fig. 21(d). We do note here that one of the small white columns along the left edge of the road in the row where the big van runs, is detected as a false change. The RS rectification methods [20], [22] perform worse than our method, even though the motion blur in this example is not significant. The change detection outputs of [20] and [22] are shown in Figs. 21(e) and (f), respectively.

In the next scenario, the side of a building is captured with camera motion that results in curvy vertical edges in

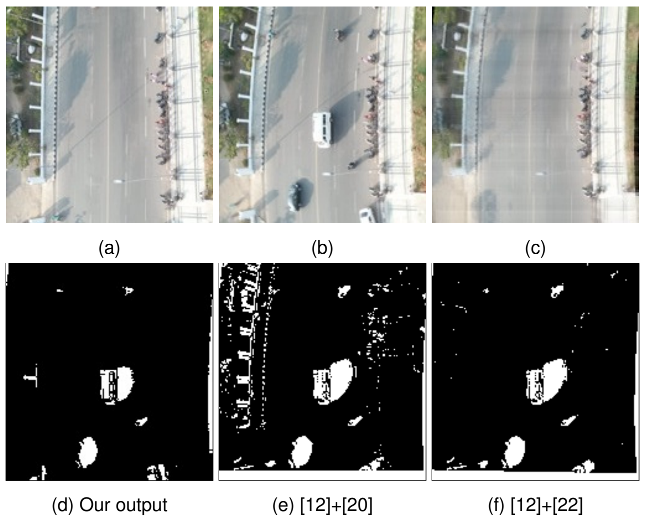


Fig. 21. Real Experiment. (a) Reference image, (b) RSMB image with prominent curves due to y-axis camera rotation, (c) Registered image, and Detected changes using (d) our method, (e) [12]+[20], and (f) [12]+[22].

Fig. 22(b) as compared to 22(a). There is a single change between the two images. Change detection should compensate for these curves and detect the new object. Our method performs well as shown in Fig. 22(d) and is better than other methods. The registered images using our method, [20] and [22] are shown in Figs. 22(c), (e), and (g), respectively. The corresponding changes are shown in Figs. 22(d), (f), and (h) respectively.

Three examples of wide area motion imagery are shown in Fig. 23. The first row shows an example with global motion blur with ground and building depth layers, and the second row shows an example with global motion blur of a planar scene. The reference image is shown in (a), the distorted image is shown in (b), the registered image is shown in (c), and the detected changes are shown in (d). After detecting changes, we remove noise by removing components smaller than a fixed size using connected component analysis. Our method performs well on both the examples, detecting even small changes.

3 Change Detection of Motion Blur-only Images

As noted earlier in Section 2.2 of the main paper, our rolling shutter motion blur model subsumes the global shutter motion blur model. In this section, we show an example for a motion blur-only scenario arising from a global shutter camera. The images are captured using a Canon EOS 60D camera. Though this camera employs CMOS sensors, the rolling shutter effect is close to negligible for common camera motion, since unlike mobile phone cameras, the readout speed is very fast. The reference image in Fig. 24(a) is captured with no camera movement, while the distorted image in Fig. 24(b) is affected with motion blur due to camera motion. All rows of the image are affected by the same camera motion. We register the image through row-wise camera pose estimation, and the corresponding registered image and detected changes are shown in Figs. 24(c) and (d), respectively. Row-wise formulation estimates

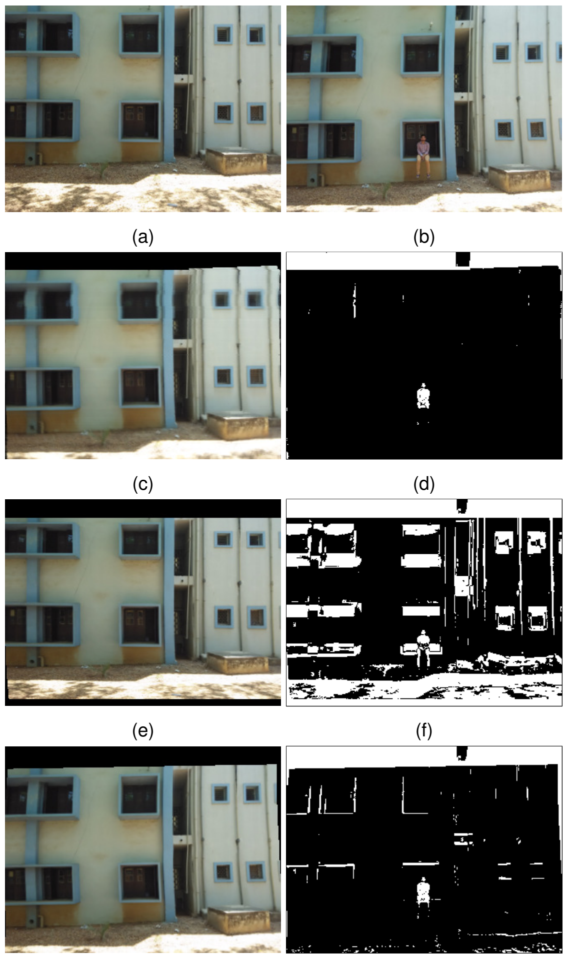


Fig. 22. Real experiment: Change detection in a 2D scene. (a) Reference image, (b) RSMB image, Registered image and detected changes using (c-d) our method, (e-f) [12]+[20], and (g-h) [12]+[22].

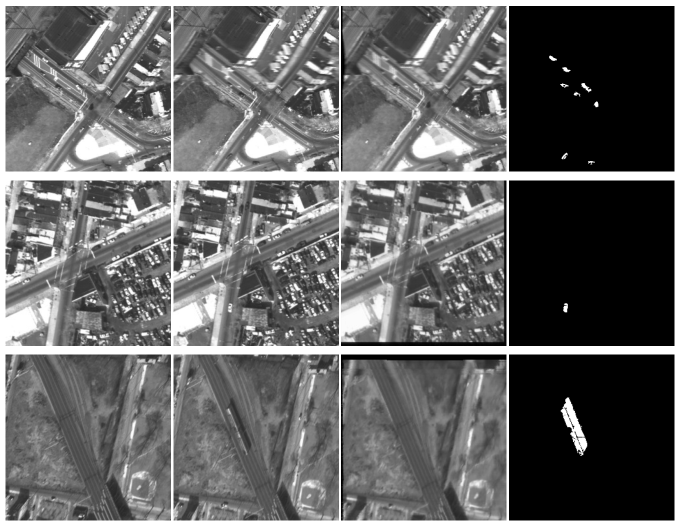


Fig. 23. Wide area motion imagery. (a) Reference image, (b) Distorted image, (c) Registered image, and (d) Detected changes.

almost same camera motion for every row, as shown in Fig. 24(e), which is indeed true for this global shutter case. With a prior knowledge of the type of distortion being global shutter motion blur, it is also possible to register the *whole* image (instead of row-wise registration) by estimating the weights for the camera poses and the change vector using (13).

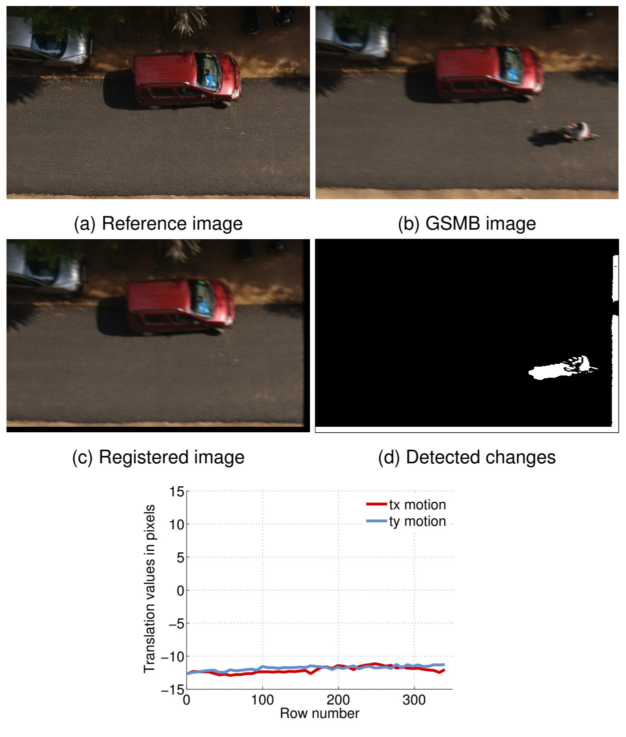
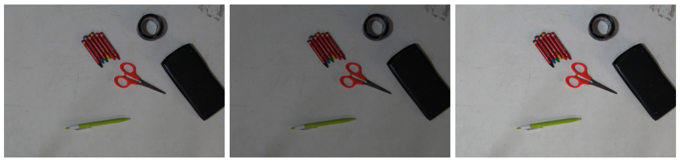


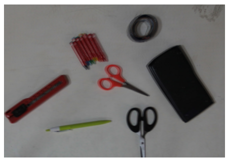
Fig. 24. Change detection in the presence of global shutter motion blur.

4 Handling Illumination Variations

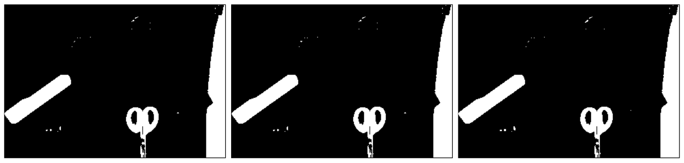
The effect of illumination variations on the performance of our method is discussed next. Since the reference and observed images are captured at different times, there could be global as well as local illumination variations. Global variations primarily involve a change in overall ambient lighting, while local variations could be due to shadows or due to the introduction and removal of one or more light sources. Our method can take care of row-wise global variations through the camera pose weight vector. Under similar illumination conditions, our optimization problem results in a camera pose weight vector summing to one due to the conservation of energy between the two images. In the case when the illumination changes globally by a multiplicative factor γ, the same vector adapts itself by summing to γ. This is demonstrated in Fig. 25. The RSMB image is shown in Fig. 25(d), and the reference images for three different illumination conditions (*same*, *low*, *high*) are, respectively, shown in Figs. 25(a), (b), and (c). Changes are correctly detected at all three illumination conditions as shown in Figs. 25(e) to (g). The sum of estimated camera pose weight vector $\|\boldsymbol{\omega}^{(i)}\|_1$ of every row for these three cases are shown in Fig. 25(h). For the *same* case, this value is close to 1 for all rows; for the *low* case, it is above 1 to compensate for the lower energy in the reference image due to a lower global illumination, and for the *high* case, it is below 1 due to a higher global illumination in the reference image.



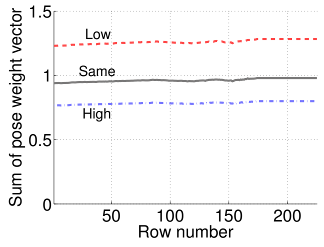
(a) *same* (b) *low* (c) *high*



(d) RSMB image



(e) *same* (f) *low* (g) *high*



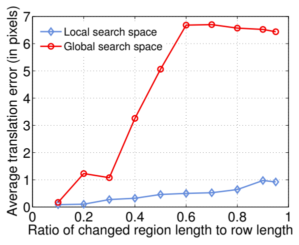
(h)

Fig. 25. Handling illumination variations. (a-c) Reference images captured under three different illumination conditions, (d) RSMB image captured under the same illumination condition as that of (a), (e-g) Detected changes, and (h) Sum of camera pose weight vector for these three cases.

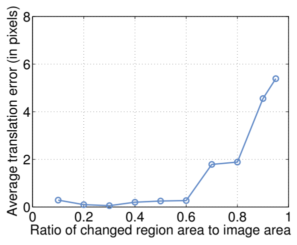
5 Effect of the Size of Changed Region

The optimisation problem in (13) involves the sparsity of the changes present in the scene. In aerial imaging and surveillance scenarios, the changes are expected to cover only a small portion of the scene, and hence our algorithm works well in these scenarios. To study this behaviour, we created a random Gaussian grayscale image f size 256×256, and added the change as a block covering ten rows. The changed region is white (with intensity 255). We then introduced horizontal translatory RSMB effect on this image. We estimate the camera motion using the clean reference image and this RSMB image. The length of the change along the row is varied to study the performance of our method. The effect of length of the changed region on the motion estimation is shown as blue line in Fig. 26(a). As the length of the changed region increases, the average pixel error increases but not very much, since we use a local neighbourhood search space $\mathcal{S}^{(i)}$ for every row. A common global search space for all rows ($\mathcal{S}^{(i)} = \mathcal{S}^{(j)}$, for all i, j) results in worse motion estimation as the size of the changed region increases; this can be seen from the red line in the same figure.

We performed similar experiments for the GSMB case. The motion is estimated using the full image instead of a row-wise estimation. The resulting estimation error for different sizes of changed regions is shown in Fig. 26(b). We can observe in this scenario too, that increasing the size of the changed region affects motion estimation. Till the area covers 60% of the image area, the estimated error is within one pixel.



(a)



(b)

Fig. 26. Effect of the size of changed regions in motion estimation in the case of (a) RSMB and (b) GSMB.

6 Evaluation Metrics

One could use a number of metrics to compare the performance of change detection result against the ground truth. Certain metrics favour certain measures; a low False Negative Rate may be favoured by a particular metric, while a low False Positive Rate may be favoured by another. In the scenario of change detection, let TP represent the number of true positives (number of pixels that are correctly detected as changes), TN represent the number of true negatives (number of pixels that are correctly detected as nonchanges), FP represent the number of false positives (number of pixels that are wrongly detected as changes), and FN represent the number of false negatives (number of pixels that are wrongly detected as nonchanges). We use the following three metrics for comparisons in Section 3.2 of the main paper: (i) *precision*, Pr = TP / (TP + FP), (ii) *percentage of wrong classification*, PWC = 100(FN + FP)/(TP+FN+FP+TN), and (iii) *F-measure*, Fm = 2 (Pr · Re)/(Pr + Re), where *recall* Re = TP / (TP + FN). Pr penalises false positives; if FP is zero, then Pr is one. PWC penalises both FP and FN; the lower the PWC, the better the performance. Fm is proportional to the harmonic mean of recall and precision; close-to-one values are preferable.