2D motion deblurring with a random camera- Analysis and simulations

Taeg Sang Cho
Computer Science and Artificial Intelligence Lab
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology, Cambridge, MA 02139

I. Introduction

In the motion invariant photography work, our main goal was to make the blur PSF invariant to object motion while making the PSF easy to invert. In this work, we pursue a rather opposite goal: we are interested in making the blur PSF as different as possible for different object motion while making the PSF easy to invert. The motivation here is analogous to the coded aperture camera. Because we make the PSF as different as possible for different object motions, we hope to easily identify the correct object motion given the input blurred image.

II. Easy to invert Vs. Easy to identify

In this section, we more rigorously define what it means for a PSF to be easily invertible and what it means to be easily identifiable. As we show in a moment, these are somewhat conflicting criteria.

A. Easy to invert

When an image blurred with an easily invertible kernel k is deblurred to give an estimate $\hat{\mathbf{x}}$ of the original sharp image $\mathbf{x}$, the squared error between $\hat{\mathbf{x}}$ and $\mathbf{x}$ (i.e. $\|\hat{\mathbf{x}}-\mathbf{x}\|^2$) is expected to be small. With an observation model $\mathbf{y}=k*\mathbf{x}+\mathbf{n}$, where $\mathbf{y}$ is the input blurry image, $\mathbf{n}$ is the observation noise, we can show that an expected squared error for all images - assuming a Gaussian image prior - is:

$$E_{\mathbf{x},\mathbf{n}}\left[\|\hat{\mathbf{x}}-\mathbf{x}\|^2\right]=\frac{1}{N}\sum_{w_x,w_y}\frac{\eta^2}{\|K(w_x,w_y)\|^2+\eta^2\Sigma^{-1}(w_x,w_y)} \tag{1}$$

where η^2 is the observation noise variance, $E_{\mathbf{x},\mathbf{n}}[\cdot]$ is an expectation of the argument with respect to $\mathbf{x}$ and $\mathbf{n}$, $\|K(w_x,w_y)\|^2$ is the Fourier transform of the blur kernel k, Σ is the covariance matrix under a Gaussian prior, and N is the dimension of the image $\mathbf{x}$. $\Sigma^{-1}(w_x,w_y)=\alpha(\|Gx(w_x,w_y)\|^2+\|Gy(w_x,w_y)\|^2)$, where $Gx(w_x,w_y)$ and $Gy(w_x,w_y)$ are the Fourier transform of $[-1,1]$ and $[-1;1]$, respectively. Note that Eq.(1) assumes that the blur kernel has been somehow correctly identified.

We normally like to work with an expected reconstruction error $E_{\mathbf{x},\mathbf{n}}\left[\|\hat{\mathbf{x}}-\mathbf{x}\|\right]$, which we define as:

$$E_{\mathbf{x},\mathbf{n}}\left[\|\hat{x}-x\|\right]=\sqrt{E_{\mathbf{x},\mathbf{n}}\left[\|\hat{x}-x\|^2\right]} \tag{2}$$

Eq.(1) tells us that in order to reduce the reconstruction error, we want $\|K(w_x,w_y)\|^2$ to be large when $\Sigma^{-1}(w_x,w_y)$ is small. Since $\Sigma^{-1}(w_x,w_y)$ is small for low frequencies, we want $\|K(u,v)\|^2$ to be as large as possible at low frequencies. The tricky rule of the game is: we want k to be easily invertible for "all" object motions of interest. While a static camera results in an easily invertible PSF for a static object, it results in a hard-to-invert PSF for moving objects because the spectrum of a box kernel has many zeros in the frequencies where $\Sigma^{-1}(w_x,w_y)$ is small. While a parabolic camera results in an easily invertible PSF for all object motions that are aligned with the camera motion, the PSF is not easily invertible for motions in other directions due to zero trenches in the spectrum. There's still some room for more improvements.

B. Easy to identify

Until now, we have assumed that the blur estimation has been done already, and ended up in a conclusion that we don't want any zeros in $\|K(w_x,w_y)\|^2$ for all object motions. However, the blur estimation is not an independent problem, and has a say in how we want $\|K(w_x,w_y)\|^2$.

Blur estimation can be formulated as the following: given a blurry image $\mathbf{y}$ and a set of kernels $k_i, i=1...M$ corresponding to different object motions, we want to find $\hat{k}$ that maximizes $p(\mathbf{y}|k_i)$. If we use a Gaussian image

prior $p(\mathbf{x})$ with zero mean and covariance matrix Σ, we can easily show that $p(\mathbf{y}|k_i) \sim N(0, K_i \Sigma K_i^T + \eta^2 I)$, where K_i is the convolution matrix of kernel k_i, and η^2 is the variance of the observation noise. Denoting $\Psi = K_i \Sigma K_i^T + \eta^2 I$,

$$\begin{aligned} \ln\left(p(\mathbf{y}|k_i)\right) &= -\frac{1}{2}\mathbf{y}^T \Psi^{-1} \mathbf{y} - \frac{1}{2}\ln\left(\det \Psi\right) + C \\ &= -\sum_{w_x, w_y} \frac{1}{2N} \frac{\Sigma^{-1}(w_x, w_y)\|Y(w_x, w_y)\|^2}{\|K_i(w_x, w_y)\|^2 + \eta^2 \Sigma^{-1}(w_x, w_y)} - \frac{1}{2}\ln\left(\det \Psi\right) + C \end{aligned} \tag{3}$$

where C is a constant independent of $\mathbf{y}$ and k_i, $Y(w_x, w_y)$ is the FT of $\mathbf{y}$, $K_i(w_x, w_y)$ is the FT of k_i, and N is the dimension of $\mathbf{y}$. Because Σ^{-1} is not positive definite, it's inverse is not defined, but with some algebraic manipulation - shown below -, we can still evaluate $\ln\left(\det \Psi\right)$.

We call $\mathbf{y}^T \Psi^{-1} \mathbf{y}$ term an estimation error, and $\ln\left(\det \Psi\right)$ term a log determinant. In order to easily identify the correct kernel, $p(\mathbf{y}|k_i)$ for wrong k_i should be much lower than $p(\mathbf{y}|\hat{k})$. How can we design k_i's in order to ensure this?

Let's first focus on the estimation error. The estimation error is evaluated with the following:

$$\sum_{u,v} \frac{1}{N} \frac{\Sigma^{-1}(w_x, w_y)\|Y(w_x, w_y)\|^2}{\|K_i(w_x, w_y)\|^2 + \eta^2 \Sigma^{-1}(w_x, w_y)} \tag{4}$$

We want this term to be as small as possible for the correct kernel $\hat{k}$, and as large as possible for all other kernels. Recall that $Y(w_x, w_y) = \hat{K}(w_x, w_y) X(w_x, w_y)$, and thus $Y(w_x, w_y)$ will have zeros at places where $\hat{K}(w_x, w_y)$ and $X(w_x, w_y)$ have zeros. Now, let's suppose that k_j is a wrong kernel, and thus we want to maximize the estimation error for this kernel. From Eq.(4), we want $K_j(w_x, w_y)$ to have zeros at frequencies $Y(w_x, w_y)$ is "not" zero, and preferably we want $\Sigma^{-1}(w_x, w_y)$ to be close to zero at those frequencies. In other words, we want the zeros in the spectrum of different kernels to lie at different frequencies.

The log determinant term biases the estimation error to favor the blur kernel with more zeros in the spectrum. This can be seen from the following relationship:

$$\begin{aligned} \ln\left(\det \Psi\right) = \ln\left(\det(K_i \Sigma K_i^T + \eta^2 I)\right) &= \sum_{w_x, w_y} \ln(\Sigma(w_x, w_y)\|K_i(w_x, w_y)\|^2 + \eta^2) \\ &= \sum_{w_x, w_y} \ln\left(\eta^2 \Sigma(u, v)(\frac{1}{\eta^2}\|K_i(w_x, w_y)\|^2 + \Sigma^{-1}(w_x, w_y))\right) \\ &= \sum_{w_x, w_y} \ln\left(\eta^2 \Sigma(w_x, w_y)\right) + \ln\left(\frac{1}{\eta^2}\|K_i(w_x, w_y)\|^2 + \Sigma^{-1}(w_x, w_y)\right) \\ &= \sum_{w_x, w_y} \ln\left(\frac{1}{\eta^2}\|K_i(w_x, w_y)\|^2 + \Sigma^{-1}(w_x, w_y)\right) + D \end{aligned} \tag{5}$$

where D is a constant independent of K. Note that $\ln\left(p(\mathbf{y}|k_i)\right)$ is increased by reducing the value of $\ln\left(\det \Psi\right)$, hence reducing $\sum_{w_x, w_y} \ln\left(\frac{1}{\eta^2}\|K_i(w_x, w_y)\|^2\right)$. Therefore, the log determinant term favors kernels with many zeros in the spectrum. This term is essentially the Occam's factor: favoring a simpler, lower capacity explanation over solutions with higher capacity. Since kernels with less zeros in the spectrum have higher capacity to capture information over those with many zeros, it makes sense that the log determinant term favors kernels with more zeros in the spectrum.

One desirable characteristic of a camera is a constant log determinant for all k_i's. This is desirable in two aspects: A constant log determinant means that the information capacity of kernels for different object motions is kept the same, thereby not favoring a single object motion; and more importantly, we can estimate the object motion solely from the estimation error in Eq.(3). Separating the solutions into different subspaces is a more reliable way to discriminate kernels than relying on the capacity argument of the kernels. While a constant log determinant term does not guarantee that the solutions of different kernels will lie in different subspaces, it's quite likely in images.

To summarize, the zeros in the spectrum are actually helping us to identify the correct blur kernels, and preferably we want those zeros at low frequencies, which conflicts with our previous result that we don't want any zeros in the

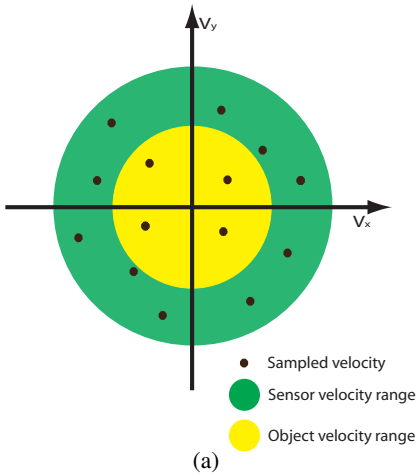


Fig. 1. This figure illustrates how **a random camera** covers the velocity space during one integration time.

low frequencies for low reconstruction error. Therefore, we should seek for a camera that covers as much frequency content as possible, while generating zeros at different frequencies for different object motions. Also, we want to find a camera that has a roughly constant log determinant for all object motions. We show that a random camera satisfies these characteristics.

III. Analysis on the Random Camera

In this work, we assume that we are interested in object speed less than S_{mo} in all directions. This is shown by the yellow velocity cover in Figure 1. Since we don't know apriori what the object velocity is, we want a camera motion that can adequately capture the object content moving in every velocity within the yellow cover. In this section, we introduce the random camera, and analyze its characteristics. In short, we show that the expected spectrum of the random camera falls off with $\frac{T}{2S_{ms}\sqrt{w_x^2+w_y^2}}$, where S_{ms} is the maximum sensor speed. This spectrum fall-off is optimal in terms of spectrum coverage, and we also show that the random camera introduces zeros differently for different object movements, which is needed for accurate object motion identification.

A. What is a random camera?

A random camera moves in a trajectory that's similar to a Brownian motion. The camera divides the integration time T into N_s segments, and moves at a constant velocity within each segment. The camera moves with different velocities in each time segment, and the way camera samples the velocity space is shown in Figure 1. The figure is in the velocity space, and the x axis denotes the speed in the x direction, and y axis denotes the speed in the y direction. The yellow region denotes the object velocity range, and the green region denotes the sensor movement velocity range.

Since we don't know apriori what the object motion is, we should sample the object velocity range (yellow region) as uniformly as possible. In most cases, we let S_{ms} be greater than S_{mo} -we will elaborate on this point in the later section.

There can be many ways to sample the velocity space as in Figure 1. Here, we assume that each segment is disconnected, and the start of each segment is randomly selected from a possible range of locations in the sensor. Figure 2 shows some PSFs corresponding to static and moving objects and their log spectrums. The number of segments (N_s) is 50. Keep in mind that the camera movement is the same for all these kernels: only the object velocity differs.

One thing to note from Figure 2 is that the envelope of the log spectrums look very similar, although the zero locations differ from one spectrum to the other. Can we predict these characteristics of a random camera?

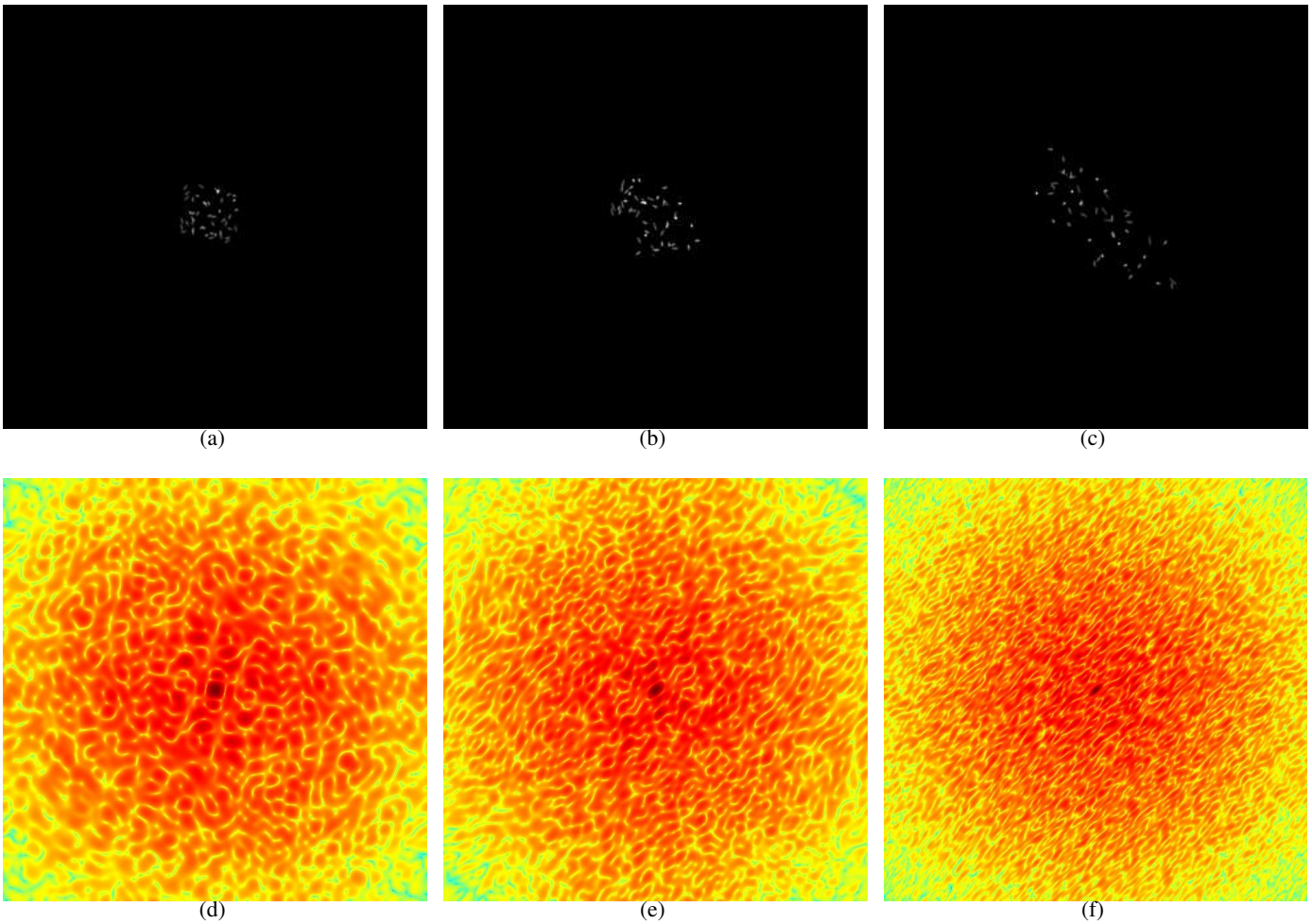


Fig. 2. N_s is 50 for this figure. Top row: the PSF of **the random camera** for a static object, for a moderate diagonal object movement, for a very fast diagonal object movement, respectively. Bottom row: the log spectrum of **the random camera** for a static object, for a moderate diagonal object movement, for a very fast diagonal object movement, respectively. The low spectrums are normalized to the same scale.

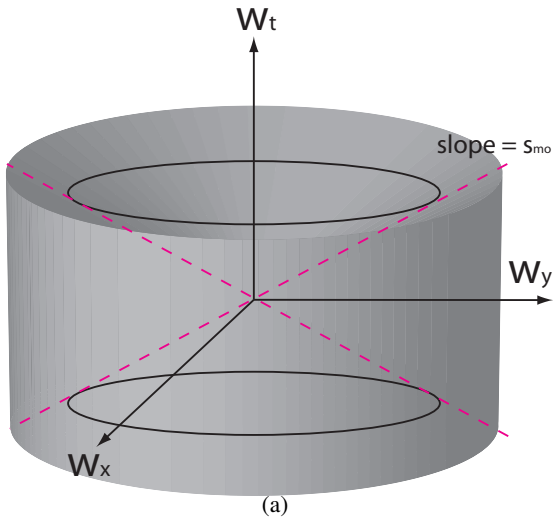


Fig. 3. This figure shows that the image content of interest all lie outside of the double cone with slope S_{mo}. Therefore, the 3D Fourier transform of the camera trajectory should be a superset of the 3D Fourier transform shown in this figure. We show in this section that the random camera optimally covers this Fourier volume.

B. Optimality

In terms of 3D Fourier transform, the image content we are interested in all lie outside of the double cone shown in Figure 3. The slope of the cone is determined by S_{mo}. To answer how this 3D Fourier transform is related to the 2D PSFs we get for different object motion, we can resort to the Fourier Slice Theorem.

The Fourier transform of a PSF for a dot moving with velocity (s_x, s_y) is a 2D slice - that passes through the origin - of the Fourier transform of the camera trajectory. Therefore, we want the 3D Fourier transform of the camera trajectory to be a super-set of the 3D Fourier transform we showed in Figure 3.

The 2D Fourier slice that corresponds to a dot moving with velocity (s_x, s_y) can be derived as the following: Let's denote the camera trajectory as $f(x, y, t)$, and the Fourier transform of f as F. When the object in the scene moves at speed (s_x, s_y), the PSF becomes:

$$\phi_{(s_x,s_y)}(x, y) = \int f(x - s_x t, y - s_y t, t)dt \tag{6}$$

Now, the Fourier transform of $\phi_{(s_x,s_y)}(x, y)$ can be computed from the following:

$$\begin{aligned}
\Phi_{(s_x,s_y)}(w_x, w_y) &= \int \phi_{(s_x,s_y)}(x, y) \exp(-j(w_x x + w_y y))dxdy \\
&= \int f(x - s_x t, y - s_y t, t) \exp(-j(w_x x + w_y y))dtdxdy \\
&= \int f(x', y', t) \exp(-j(w_x x' + w_y y' + (w_x s_x + w_y s_y)t))dtdx'dy' \\
&= F(w_x, w_y, w_x s_x + w_y s_y)
\end{aligned} \tag{7}$$

Therefore, the FT of $\phi_{(s_x,s_y)}(x, y)$ is a slice along $F(w_x, w_y, w_x s_x + w_y s_y)$. In other words, as the x-directional speed changes, the slice rotates about w_y axis, and as the y-directional speed changes, the slice rotates about w_x axis.

We can follow the conservation of energy argument in the motion invariant photography paper to show that the optimal fall-off in the 2D $w_x - w_y$ slice of the 3D power spectrum of the 3D space-time camera trajectory should be proportional to $\frac{T}{2S_{ms}\sqrt{w_x^2+w_y^2}}$. Notice that this spectrum has the same shape as in Figure 3! We can show that if $S_{ms} \to \infty$, the random camera in expectation achieves the optimal fall-off, regardless of N_s, the number of segments during the integration time.

To show this, we consider a 2D version of the problem with the space-time trajectory shown in Figure 4. The black lines denote the camera trajectory, and the red blocks denote the projected trajectory (i.e. the 1D PSF.) The slope of each black lines is the speed of the camera, and the width of the PSF that corresponds to this black line is a function of that speed. For a trajectory with infinite speed, the black line would be horizontal.

We can show that the width l of the projected box filter is $S_{seg}\frac{T}{N_s}$, where S_{seg} is the speed within the segment, by noting that the length of the time segment is fixed to $\frac{T}{N_s}$, and $S_{seg} = \tan(\phi) = \frac{l}{\frac{T}{N_s}}$. The PSF is just a sum of all the projected box filters with corresponding lengths:

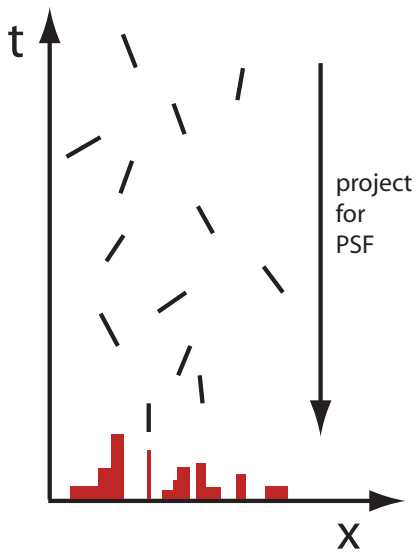
$$k(x) = \sum_i b_i(x) \tag{8}$$

where $b_i(x)$ is a box filter that corresponds to the projection of i^{th} segment. The power spectrum of k can be shown to be:

$$\|K(w_x)\|^2 = \sum_i \|B_i(w_x)\|^2 + \sum_{i \neq j} B_i^*(w_x)B_j(w_x) \tag{9}$$

where $B_i(w_x)$ is the Fourier transform of $b_i(x)$, and * is a complex conjugate. $\sum_{i \neq j} B_i^*(w_x)B_j(w_x)$ manifests itself as a constructive/destructive addition of spectrum to $\sum_i \|B_i(w_x)\|^2$ depending on the phase between $B_i^*(w_x)$ and $B_j(w_x)$.

Since the camera speed within each segment is randomly selected within the green velocity cover in Figure 1, we want to compute the expected power spectrum of the kernel with respect to the segment speed. The expected



(a)

Fig. 4. 2D space-time trajectory of **a random camera**, and the projected PSF (shown by red blocks.)

power spectrum of k is:

$$\begin{aligned} E_s[\|K(w_x)\|^2] &= E_s[\sum_i \|B_i(w_x)\|^2 + \sum_{i\neq j} B_i^*(w_x)B_j(w_x)] \\ &= \sum_i E_s[\|B_i(w_x)\|^2] + E_s[\sum_{i\neq j} B_i^*(w_x)B_j(w_x)] \end{aligned} \tag{10}$$

Since each segments are made statistically independent by randomly choosing velocities for each segment as well as the starting position of each segment,

$$E_s[\sum_{i\neq j} B_i^*(w_x)B_j(w_x)] = \sum_{i\neq j} E_s[B_i^*(w_x)]E_s[B_j(w_x)] = \mu(w_x)^2 \tag{11}$$

If the kernel has a large support, $\mu(w_x) \approx \delta(w_x)$ and does not affect the spectrum at high frequencies.

To compute $E_s[\|B_i(w_x)\|^2]$, note that a box filter in the spatial domain is a sinc in the frequency domain. Therefore, the expected power spectrum of individual segment (assuming that a uniform probability distribution on s) becomes:

$$E_s[\|B_i(w_x)\|^2] = \left(\frac{1}{2S_{ms}}\int_{-S_{ms}}^{S_{ms}} \epsilon^2 sinc^2(\epsilon s w_x) ds\right) \tag{12}$$

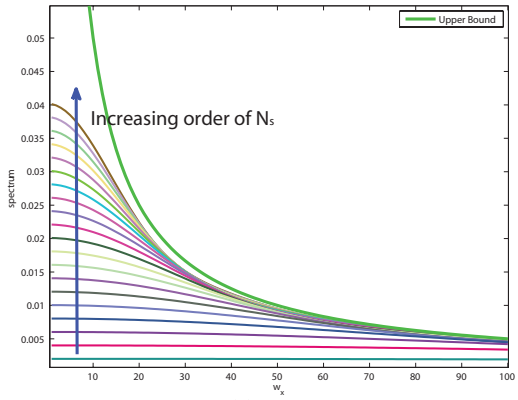
where $\epsilon = \frac{T}{N_s}$. Therefore, $\sum_i E_s[\|B_i(w_x)\|^2]$ becomes:

$$\sum_i E_s[\|B_i(w_x)\|^2] = \frac{T}{\epsilon}\left(\frac{1}{2S_{ms}}\int_{-S_{ms}}^{S_{ms}} \epsilon^2 sinc^2(\epsilon s w_x) ds\right) \tag{13}$$

As we let S_{ms} increase indefinitely, $\sum_i E_s[\|B_i(w_x)\|^2]$ approaches $\frac{T}{2S_{ms}w_x}$, independent of ϵ. In other words, if $\sum_{i\neq j} E_s[B_i^*(w_x)B_j(w_x)]$ is sufficiently small for all frequencies (except for $w_x = 0$), the power spectrum of the random camera falls off proportional to $\frac{T}{2S_{ms}w_x}$ as $S_{ms} \to \infty$. We can extend this result to the 3D space-time trajectory as well to show that the $w_x - w_y$ slice of the random camera's 3D spectrum falls off proportional to $\frac{T}{2S_{ms}\sqrt{w_x^2+w_y^2}}$.

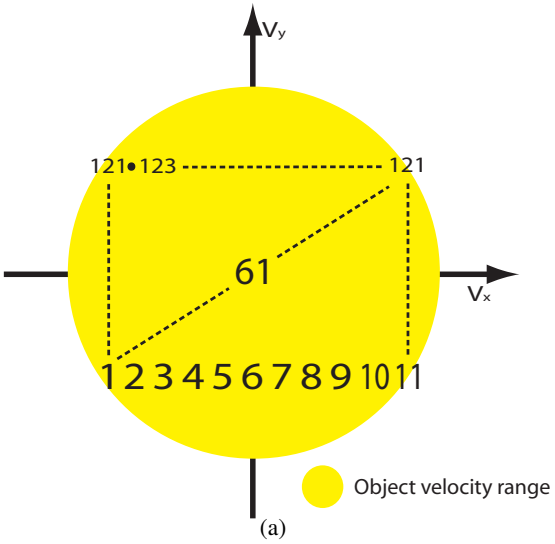
For practical reasons, S_{ms} cannot be made ∞, so we want to see how $E_s[\|K(w_x)\|^2]$ behaves with finite S_{ms}. We can in fact analytically compute $E_s[\|K(w_x)\|^2]$ with finite S_{ms}:

$$E_s[\|K(w_x)\|^2] = \frac{T\epsilon}{S_{ms}^2\epsilon^2\pi^2 w_x^2}\left(-0.5 + 0.5\cos(2\pi\epsilon S_{ms}w_x) + (\pi\epsilon S_{ms}w_x)\int_0^{2\pi\epsilon S_{ms}w_x} \frac{sin(x)}{x} dx\right) \tag{14}$$



(a)

Fig. 5. The green curve at the top is the upper bound derived for $S_{ms} \rightarrow \infty$, and the curves at the bottom correspond to $E_s[\|K(w_x)\|^2]$ for different ϵ.



(a)

Fig. 6. 121 motions considered in this report. The numbers on the $v_x - v_y$ axis denote the motion numbers: motion 1 denotes a very fast diagonal object movement, motion 61 denotes a static object movement etc... For each horizontal speed, there are 11 different vertical speeds, and vice versa.

To see how $E_s[\|K(w_x)\|^2]$ with finite S_{ms} behaves with finite number of velocity samples, we set S_{ms} to be the value used in the experiments, and vary ϵ. We increase ϵ in $\frac{T}{N_s}$ steps, where $N_s = 20$, and compute $E_s[\|K(w_x)\|^2]$. The computed spectrum is plotted in Figure 5. The green curve at the top is $\frac{1}{w_x}$ for $S_{ms} \rightarrow \infty$, and the curves at the bottom correspond to $\frac{S_{ms}}{T} E_s[\|K(w_x)\|^2]$ for different ϵ. One thing to notice is that as we decrease ϵ (i.e. increase the number of segments) the spectrum approximates $\frac{1}{w_x}$ bound better. In other words, with finite S_{ms}, the number of random segments within the integration time does matter, and to approach the spectrum proportional to $\frac{1}{w_x}$ we need to increase the number of random segments within the integration time as much as possible. As we increase S_{ms} we need less number of segments to approach the spectrum proportional to $\frac{1}{w_x}$, but then we would be spreading the spectrum for larger set of velocities, which is undesirable.

Until now we only considered a PSF of a static object. In the case of a moving object, as long as the maximum sensor velocity (S_{ms}) is large enough to make the effective velocity contribution from the object irrelevant, the above results still hold.

In short, there are two important parameters for a random camera: S_{ms} and N_s. We will have to search over these parameters to find a practical camera.

C. The weighted reconstruction error and the log determinants

In this section, we show the weighted reconstruction error, the estimation error and the log determinants of a random camera for different object motions. In the earlier section, we showed that the reconstruction error can be computed with the following relationship:

$$E_{\mathbf{x},\mathbf{n}}\left[\|\hat{\mathbf{x}}-\mathbf{x}\|^2\right] = \frac{1}{N}\sum_{w_x,w_y}\frac{\eta^2}{\|K(w_x,w_y)\|^2+\eta^2\Sigma^{-1}(w_x,w_y)} \tag{15}$$

Because we assumed the Gaussian image prior when deriving Eq.(15), the reconstruction error measure favors kernels with large low-frequency Fourier coefficients. However, we are more interested in favoring kernels that can capture high frequency contents (we are interested in capturing edges!). Therefore, we modify the reconstruction error term by adding a weight that's proportional to the frequency:

$$E_{\mathbf{x},\mathbf{n}}\left[\|\hat{\mathbf{x}}-\mathbf{x}\|\right] = \sqrt{\frac{1}{N}\sum_{w_x,w_y}\left(\sqrt{w_x^2+w_y^2}\right)\frac{\eta^2}{\|K(w_x,w_y)\|^2+\eta^2\Sigma^{-1}(w_x,w_y)}} \tag{16}$$

We consider 121 different motions, each numbered as in Figure 6. For each horizontal speed, there are 11 different vertical speeds, and vice versa. We fix N_s to be 50, $S_{ms} = 2 \times S_{mo}$, and sample 5 different camera movements (i.e. 5 different sets of velocities to be covered.) We plot the weighted reconstruction error and the log determinant of the kernels corresponding to different motions in Figure 7(a, d).

One thing to notice is that the weighted reconstruction error plots for 5 different camera movement samples are quite flat. Also, as we expect, the log determinant plots are quite flat as well. We compared the weighted reconstruction error and the log determinants with that of a static camera and a vertical parabolic camera, shown in Figure 7(b, c, e, f). Note the scale difference. The variation of these measures in a static camera is much larger than that of a random camera: while the weighted reconstruction error performance is extremely good for a static object (motion 61), the performance degrades quickly for other motions, especially fast ones. Also, the variation of these measures in a parabolic camera is greater than that of random cameras. When the parabolic camera movement is aligned with the object movement, the parabolic camera does a great job capturing the image content, but when the camera and the object movements are not aligned, the performance degrades.

Interestingly, the parabolic camera does quite well even if is not aligned with the object motion. In fact, when we look at the Fourier spectrum of a PSF when the camera is not aligned with the motion, the zero trench of the PSF's spectrum is not completely zero, and there's spectrum spillage into the wedge that helps in reconstructing those frequencies. Also, the angle of the wedge is less than 45^o even when the object moves at its maximum orthogonal speed so the spectrum loss is quite small. On the other hand, the zeros in the spectrum of random cameras degrade the weighted reconstruction performance more than we expected. We show later that the correct PSF estimation with parabolic cameras is extremely hard, and the overall performance is better with the random camera. If we take the average spectrum of multiple random kernels (to approximate the expectation), the weighted reconstruction is better with the random camera than with the single parabolic camera.

We want to see how flat the log determinant of a random camera is compared with its estimation error. If the variation of the log determinant is significantly smaller than that of the estimation error, we may be able to estimate the PSF using only the estimation error. Figure 8 shows the estimation error, the log determinant, and the sum of the two for 3 different object motions. We can immediately see that the variation in the log determinant term is insignificant compared to the variation in the estimation error term. This suggests that the PSF estimation could be done solely with the estimation error term.

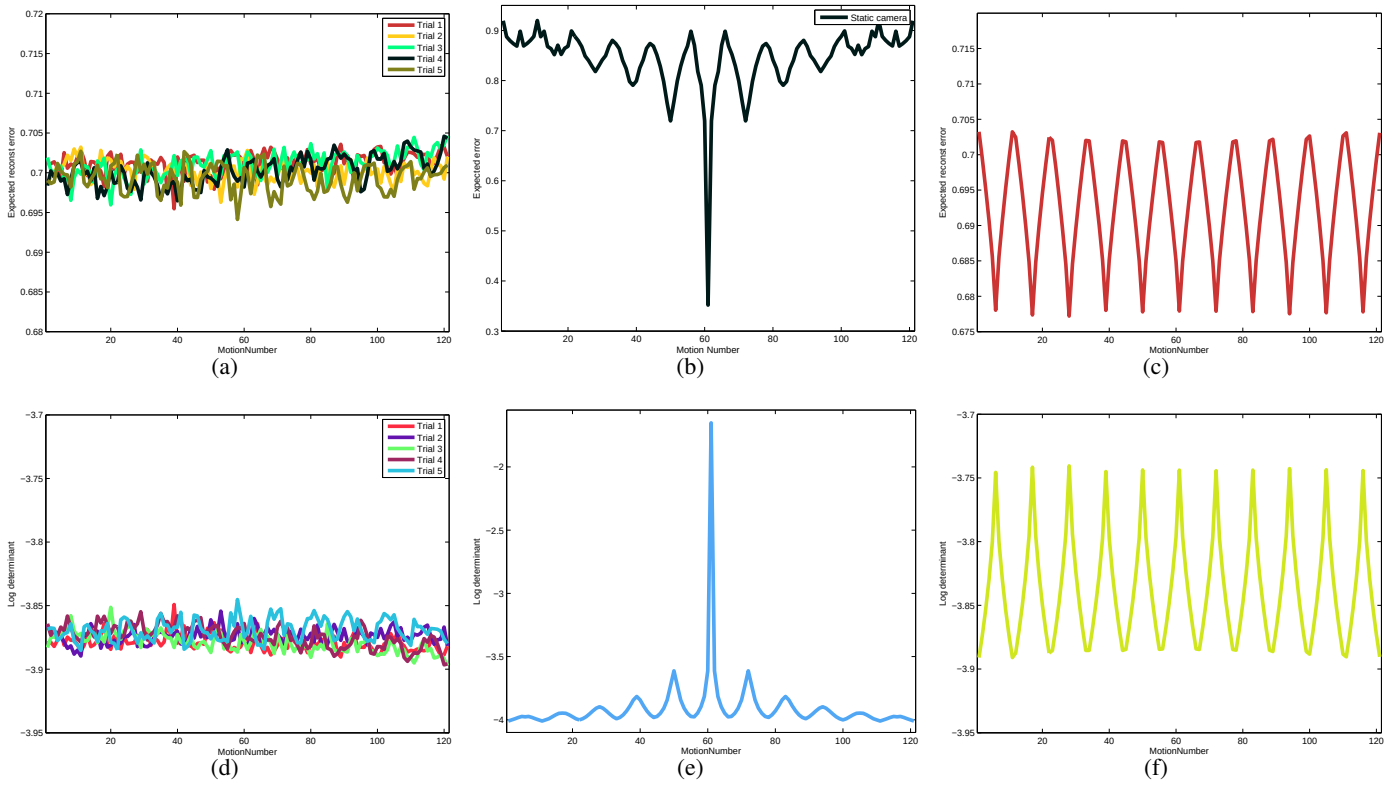


Fig. 7. (a) Weighted reconstruction errors of **a random camera** when $N_s = 50$. (b) The weighted reconstruction error of **a static camera**. (c) The weighted reconstruction error of **a vertical parabolic camera**. (d) Log determinants of **a random camera** when $N_s = 50$. (e) The log determinants of **a static camera**. (f) The log determinants of **a vertical parabolic camera**. Note the scale difference in these plots.

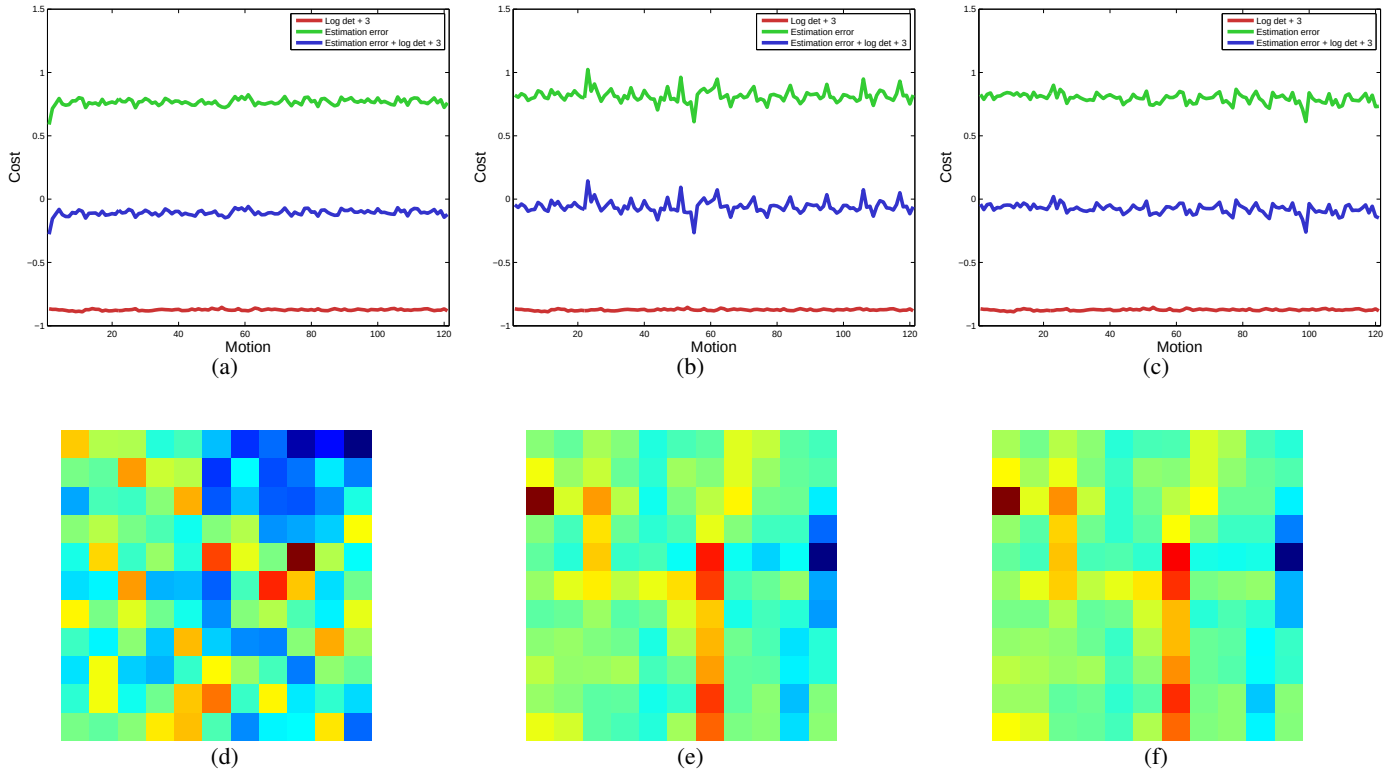


Fig. 8. (a) The estimation error, the log determinant, and the sum of the two for a very fast diagonal object motion (motion 1.) (b) The estimation error, the log determinant, and the sum of the two for a slow horizontal object motion (motion 55.) (c) The estimation error, the log determinant, and the sum of the two for a diagonal object motion (motion 99.) (d) The log determinant normalized to fit [0,1] scale and rearranged to a 11 x 11 motion grid. (e) The estimation error in (b) normalized to fit [0,1] scale and rearranged to a 11 x 11 motion grid. (f) The sum of unnormalized log determinant and the estimation error in (b) normalized and rearranged to a 11 x 11 motion grid.

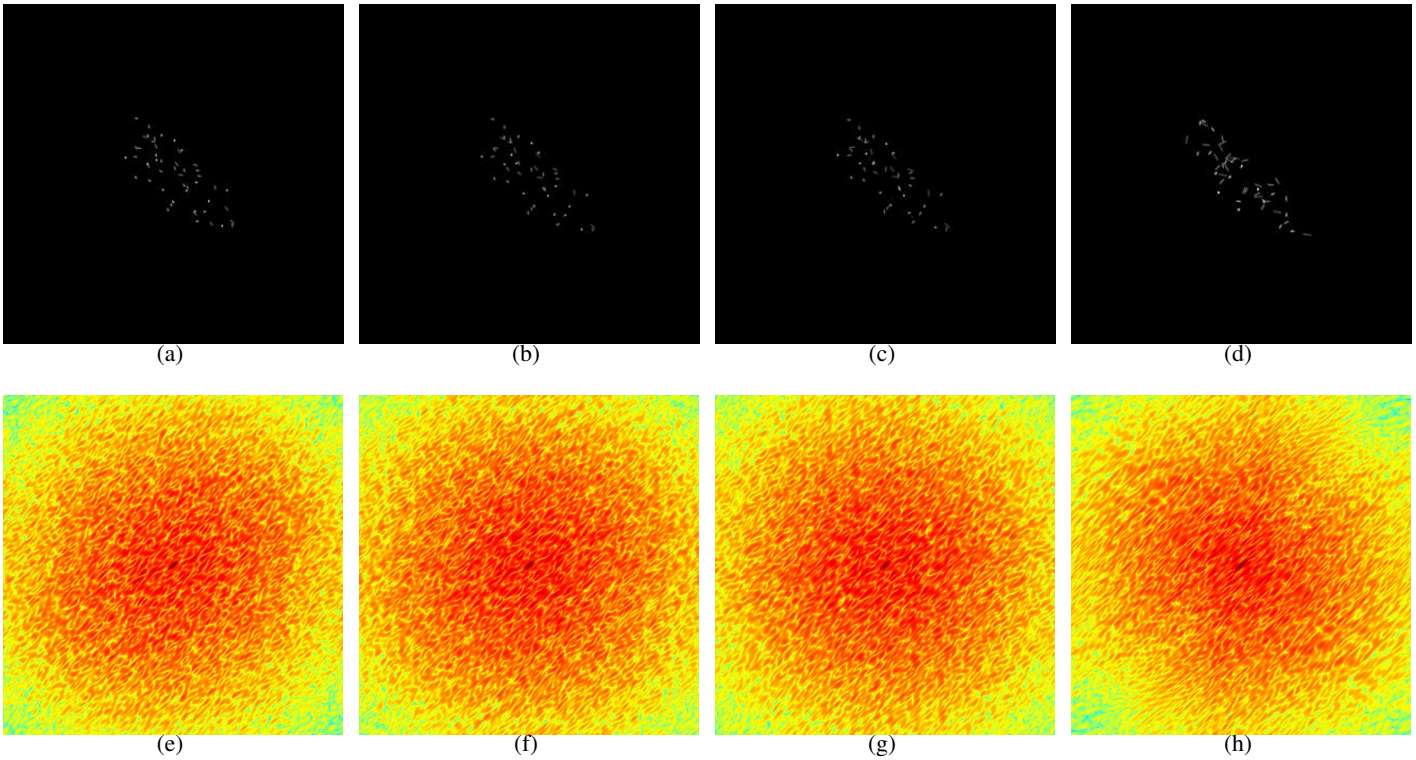


Fig. 9. Top row: the PSF of **the random camera** for motion 1 (very fast diagonal) when (a) $S_{ms} = 1.2 \times S_{mo}$, (b) $S_{ms} = 1.5 \times S_{mo}$, (c) $S_{ms} = 2 \times S_{mo}$, (d) $S_{ms} = 3 \times S_{mo}$. Bottom row: the log spectrum of the kernels in the top row.

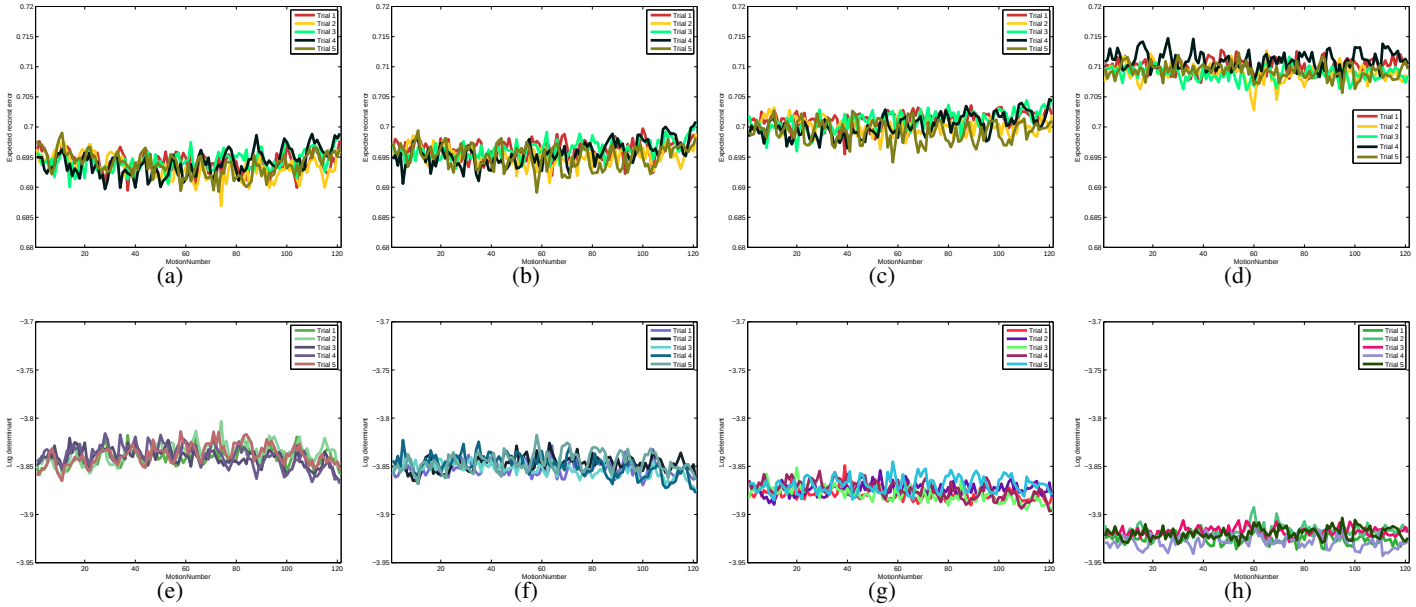


Fig. 10. Top row: The weighted reconstruction error of **the random camera** when (a) $S_{ms} = 1.2 \times S_{mo}$, (b) $S_{ms} = 1.5 \times S_{mo}$, (c) $S_{ms} = 2 \times S_{mo}$, (d) $S_{ms} = 3 \times S_{mo}$. Bottom row: the log determinant of the random camera when (e) $S_{ms} = 1.2 \times S_{mo}$, (f) $S_{ms} = 1.5 \times S_{mo}$, (g) $S_{ms} = 2 \times S_{mo}$, (h) $S_{ms} = 3 \times S_{mo}$.

D. The performance dependence on S_{ms} and N_s

In the previous section, we fixed S_{ms} and N_s in computing the weighted reconstruction error and the log determinant. In this section, we want to analyze how S_{ms} and N_s affect the performance of a random camera. We can anticipate that as we increase S_{ms}, the "average" (over object motions) weighted reconstruction error will go up (since the spectral power is shared among larger range of velocities), but the variation of the weighted reconstruction

error among different object motions will decrease, especially for fast object motions. As we increase N_s, the variation of the weighted reconstruction error among different object motions will decrease, and also the "average" (over object motions) weighted reconstruction error will decrease. However, we cannot indefinitely increase N_s due to lens blur, finite sensor resolution, physical implementation constraints etc...

1) Sweeping S_{ms}: To verify the performance dependence on S_{ms}, we vary the value of S_{ms} and plot the weighted reconstruction error and the log determinant. Again, for each S_{ms} value we sampled 4 different camera movements. Values of S_{ms} considered are $1.2 \times S_{mo}$, $1.5 \times S_{mo}$, $2 \times S_{mo}$, and $3 \times S_{mo}$. We fix N_s to 50 in this section. The noise variance is assumed to be 0.01. Figure 9 shows some random PSF's and their respective log spectrums for a fast diagonal object movement. The log spectrums all fall off proportional to the distance from the origin, and have distinguishable zero structures.

It's hard to quantify the characteristic just by looking at the log spectrums. So we plot the weighted reconstruction error and the log determinants for the random camera in Figure 10. One evident feature in the weighted reconstruction error is that as we increase S_{ms}, the average weighted reconstruction error increases as well. The variation in the weighted reconstruction error doesn't seem to change much as we sweep S_{ms}: the variation of the weighted reconstruction error is more strongly affected by N_s once S_{ms} sufficiently large. Another reason for the minor effect of S_{ms} is that each segments are all disconnected. The reason for increasing S_{ms} is that the kernel support (or the randomness in the kernel) does not get affected by the object movement. If the camera trace is all disconnected, it's more likely that the kernel support will be less affected by the object movement than the case where the camera trace is all connected. We will explore this issue more in the later section.

2) Sweeping N_s: To verify the performance dependence on N_s, we vary the value of N_s and plot the weighted reconstruction error and the log determinant. Again, for each N_s value we sampled 5 different camera movements. Values of N_s considered are 10, 30, 50, and 70. We fix S_{ms} to $2 \times S_{mo}$ in this section. The noise variance is assumed to be 0.01. Figure 11 shows some random PSF's and their respective log spectrums for a fast diagonal object movement. When N_s is small, we can observe the elongated directional structure in the spectrum, and the zero structures are quite large. As we increase N_s, the elongated structure in the spectrum disappears, and the zero structures become much finer since more segments interact to give constructive/destructive phase coherence.

The weighted reconstruction error and the log determinant plots as we sweep N_s are shown in Figure 12. As we increase N_s, the variation in the weighted reconstruction error and the log determinants becomes smaller, but the average value of the weighted reconstruction error slightly increases. While this increase in the weighted reconstruction error is somewhat unexpected, it can be attributed to finer zeros in the spectrum.

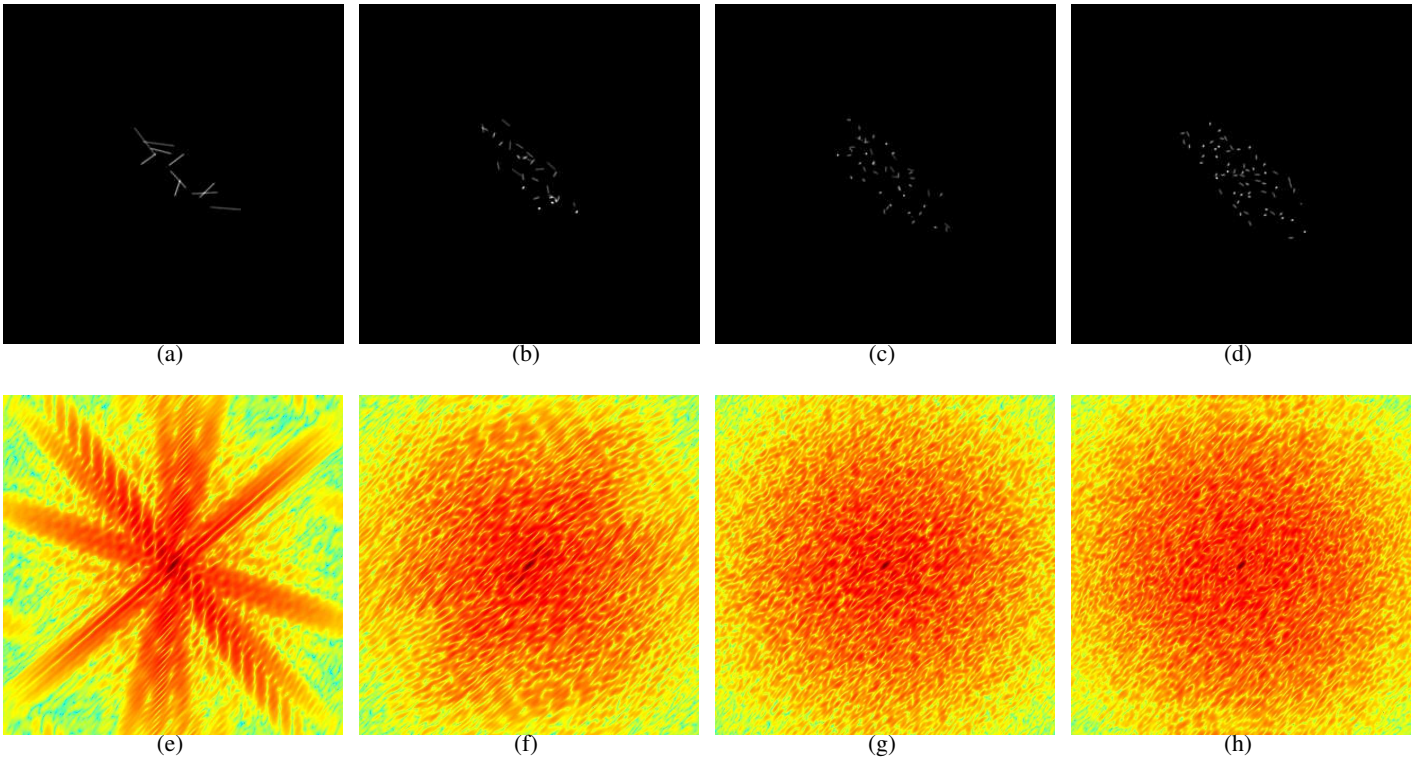


Fig. 11. Top row: the PSF of **the random camera** for motion 1 (very fast diagonal) when (a) $N_s = 10$, (b) $N_s = 30$, (c) $N_s = 50$, (d) $N_s = 70$. Bottom row: the log spectrum of the kernels in the top row.

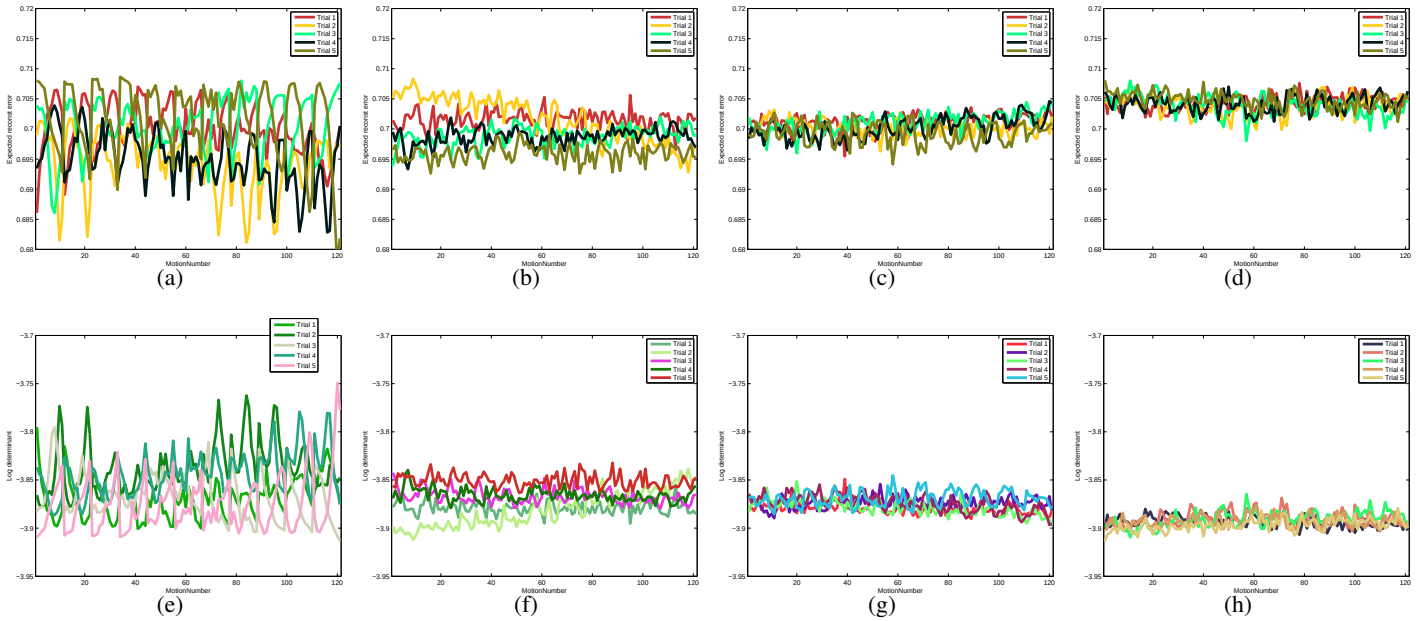
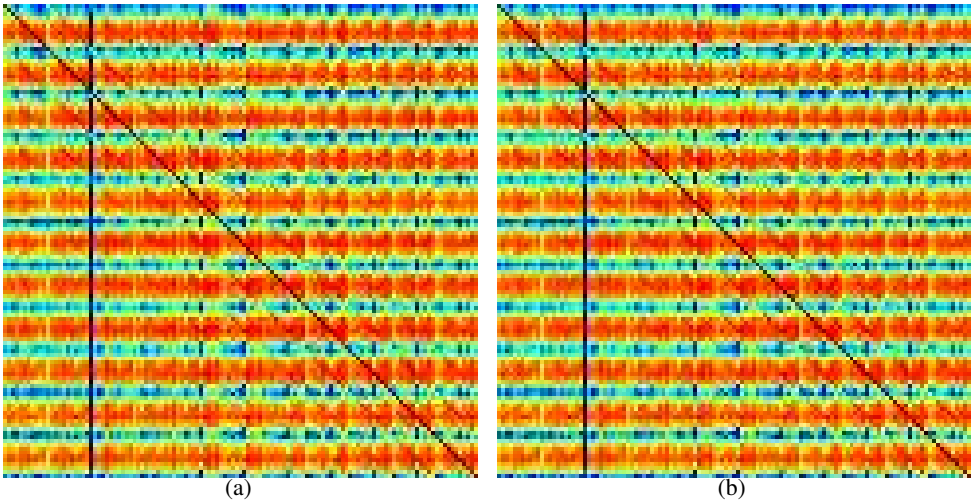


Fig. 12. Top row: the weighted reconstruction error of **the random camera** when (a) $N_s = 10$, (b) $N_s = 30$, (c) $N_s = 50$, (d) $N_s = 70$. Bottom row: the log determinant of the random camera when when (e) $N_s = 10$, (f) $N_s = 30$, (g) $N_s = 50$, (h) $N_s = 70$.



(a) (b)

Fig. 13. Red signifies a high value, and blue signifies a low value. (a) A confusion matrix using the estimation error measure. (b) A confusion matrix using the log likelihood measure

E. Identifying the PSF from the input image

The analysis from the previous section taught us that it's reasonable to make $S_{ms} = 2 \times S_{mo}$, and make $N_s = 50$. We will now examine how this parameter setting performs in estimating the PSF from the input image. We use both the full log likelihood Eq.(3) and the estimation error Eq.(4) to estimate the PSF. We consider two scenarios in the PSF estimation: the whole image has been blurred with the same kernel; different parts of the image have been blurred with different blur kernels.

1) PSF estimation using the whole image: We estimate the PSF assuming a single blur kernel across the whole image. 1% noise was added to all blurred images. The image size is in the range of 600 x 800 for the next two examples. The PSF estimation using the entire image is an easier problem than trying to estimate the PSF using local patches since we have more data samples to average over. We will explore the PSF estimation using local patches in the next section.

In this setup, we get 100% PSF estimation performance using both the estimation error based measure and the full log likelihood based measure. We plot the confusion matrices for the estimation error based classification and the log likelihood based classification in Figure 13. The confusion matrix is row-wise normalized. The confusion matrix is 121 x 121, and the (i, j) entry is the score given to the hypothesis that the image originally blurred with i^{th} kernel is identified as being blurred with j^{th} kernel. Higher score means that the algorithm thinks it's more likely. Some slices of these confusion matrices appeared in Figure 8. As you can see from Figure 8, the estimation error alone gets the PSF estimation correct!

2) PSF estimation using local patches: In this paragraph, we consider the PSF estimation using small localized neighborhoods. This allows us to detect different motions within the same image. The resolution of the estimated layer can be increased by reducing the size of the neighborhood (i.e. the patch size), but we cannot indefinitely decrease the patch size because we need enough data samples to make the decisions. Therefore, we set the patch size to be 41 x 41. 1% noise was added to all blurred images.

We use both the estimation based measure and the full log likelihood based measure to locally estimate the PSF in two images. Each image was blurred with the same kernel across the whole image. For each blurry image, we compare the reconstructed image with that of the static camera and the parabolic camera.

There are a few points to make in this section. First, the quality of the reconstructed image knowing the correct PSF using a random camera (Figure 14(d), Figure 17(d)) is much better than that of a static camera (Figure 15(d), Figure 18(d)) and similar to a parabolic camera (Figure 16(d), Figure 19(d)). This was expected from the weighted reconstruction error. Secondly, the PSF estimation performance of the random camera also surpasses that of the static camera and the parabolic camera. With the random camera, the estimation error is enough to accurately estimate the blur kernel (Figure 14(b)), but the estimation error always favors a no motion explanation in the static camera (Figure 15(e)). Log determinant does help in estimating the PSF for the static camera case, but even with

the log determinant, the blur kernel estimation performance (Figure 15(f)) is worse than that of a random camera (Figure 15(c)). Interestingly, for the static camera, the log likelihood based measure still cannot distinguish two motions moving at the same velocity, but in different directions. Such misclassification does not result in artifacts, though, since the blur kernel looks the same in those two cases.

With the parabolic camera, the PSF estimation pretty much does not work. One of the reasons for this failure is that there is too much ambiguity among kernels in the same directions. Another type of ambiguity comes from kernels that corresponds to different orthogonal speeds. When there's no orthogonal movement (say y directional movement) in the object, there isn't any zero trench in the w_y axis, and the Fourier transform will have a long elongated structure in the w_y axis. When there's a y-directional movement, however, you will see a zero trench in the w_y axis, and the ambiguity comes from the fact that the kernel without zero trench is always a superset of a kernel with the zero trench in the w_y axis, and the ambiguity among them can only be resolved through the log determinant term. In other words, the solutions lie on the same subspace in the case of parabolic cameras that it's hard to separate them out.

One downside of a random camera is that when the PSF estimation is wrong, the reconstructed image contains visible boundaries. The main reason for such visible boundaries is that the center point of a blur kernel drifts as the object motion changes. In other words, the kernel is not symmetric about the origin. This is different from a static camera since the center point of all blur kernels is the same. In order to reduce such artifacts, we may have to apply a graph-cut based blur layer smoothing to reduce the kernel estimation error before generating the deblurred image.

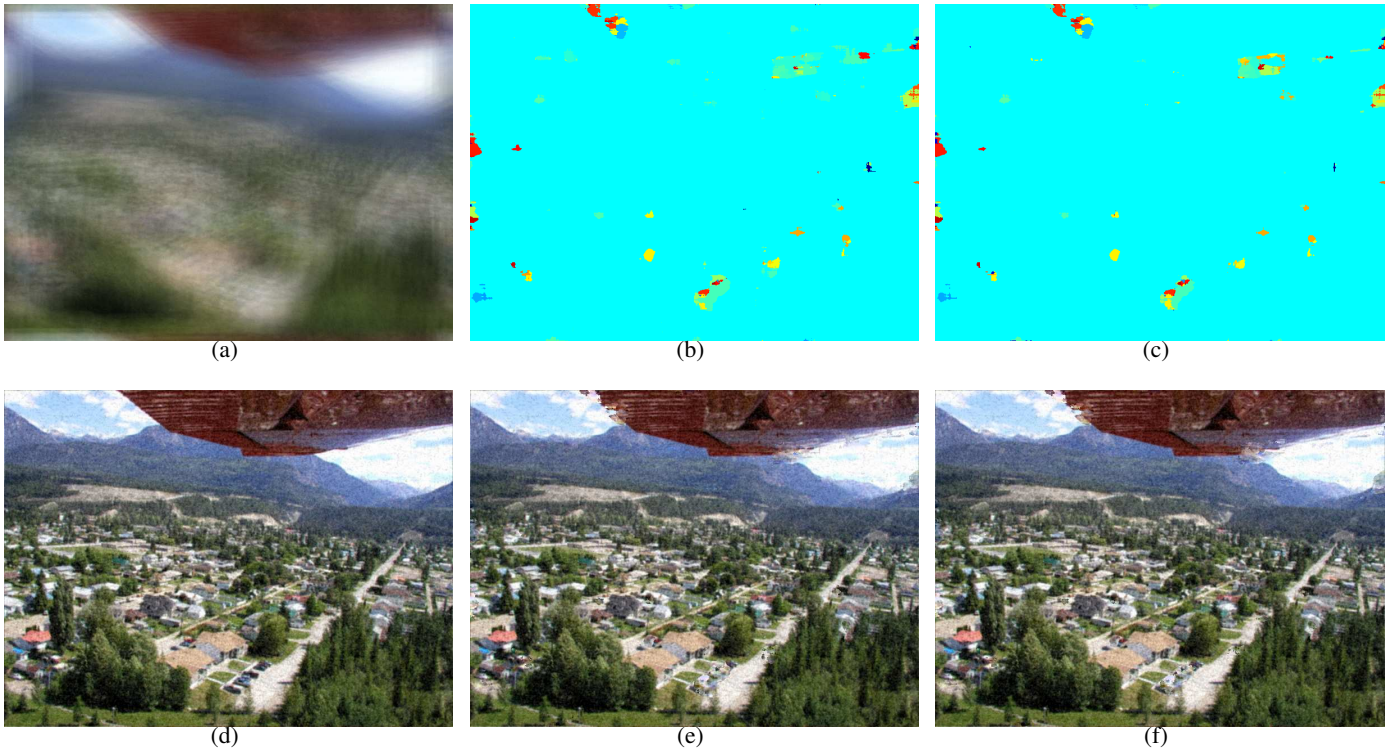


Fig. 14. (a) Input blurry image taken with **a random camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

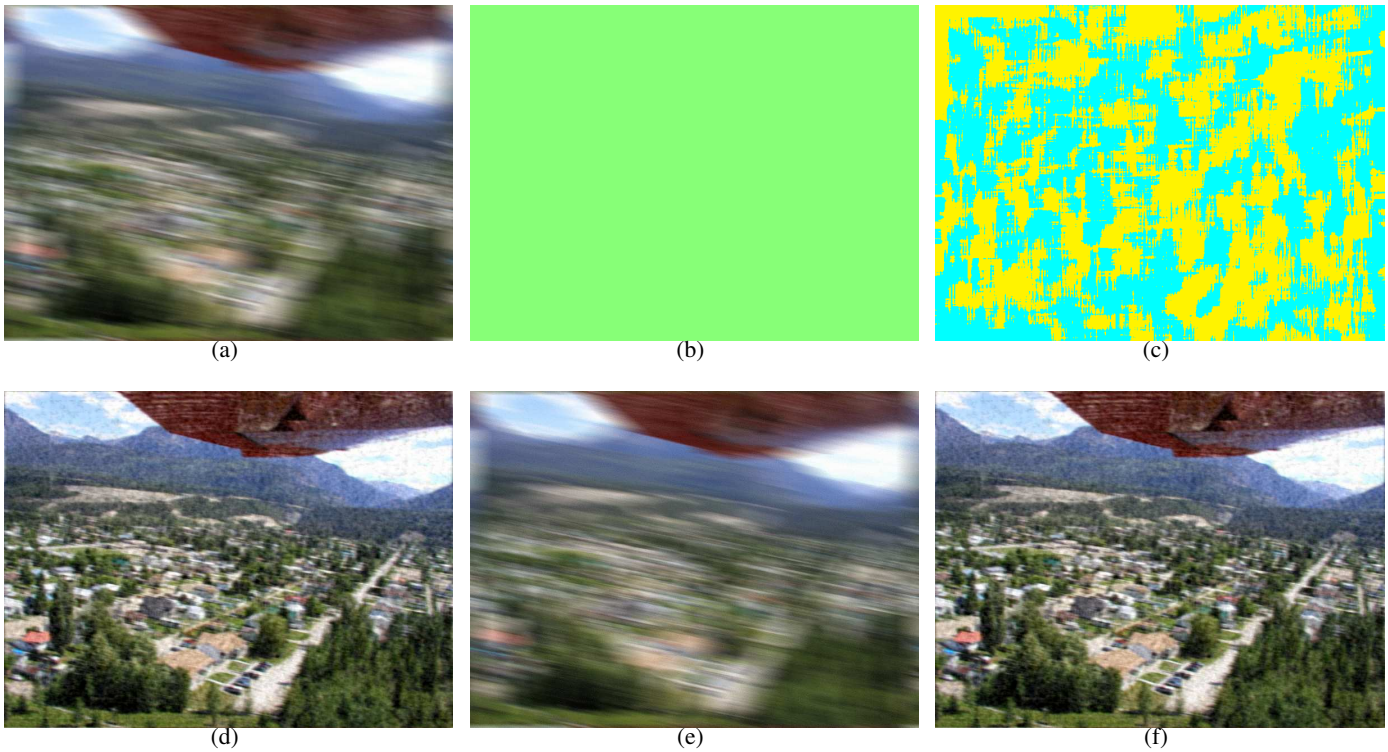
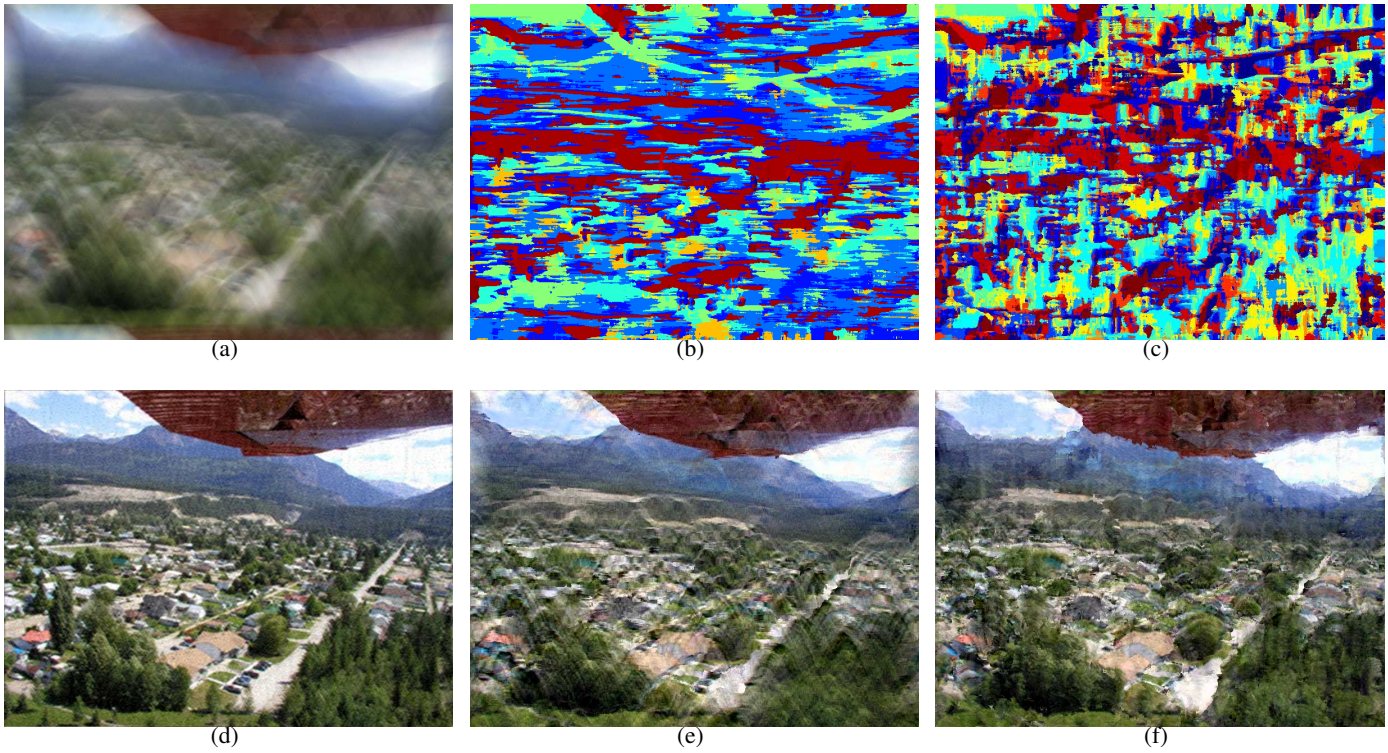


Fig. 15. (a) Input blurry image taken with **a static camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

Fig. 16. (a) Input blurry image taken with **a vertical parabolic camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

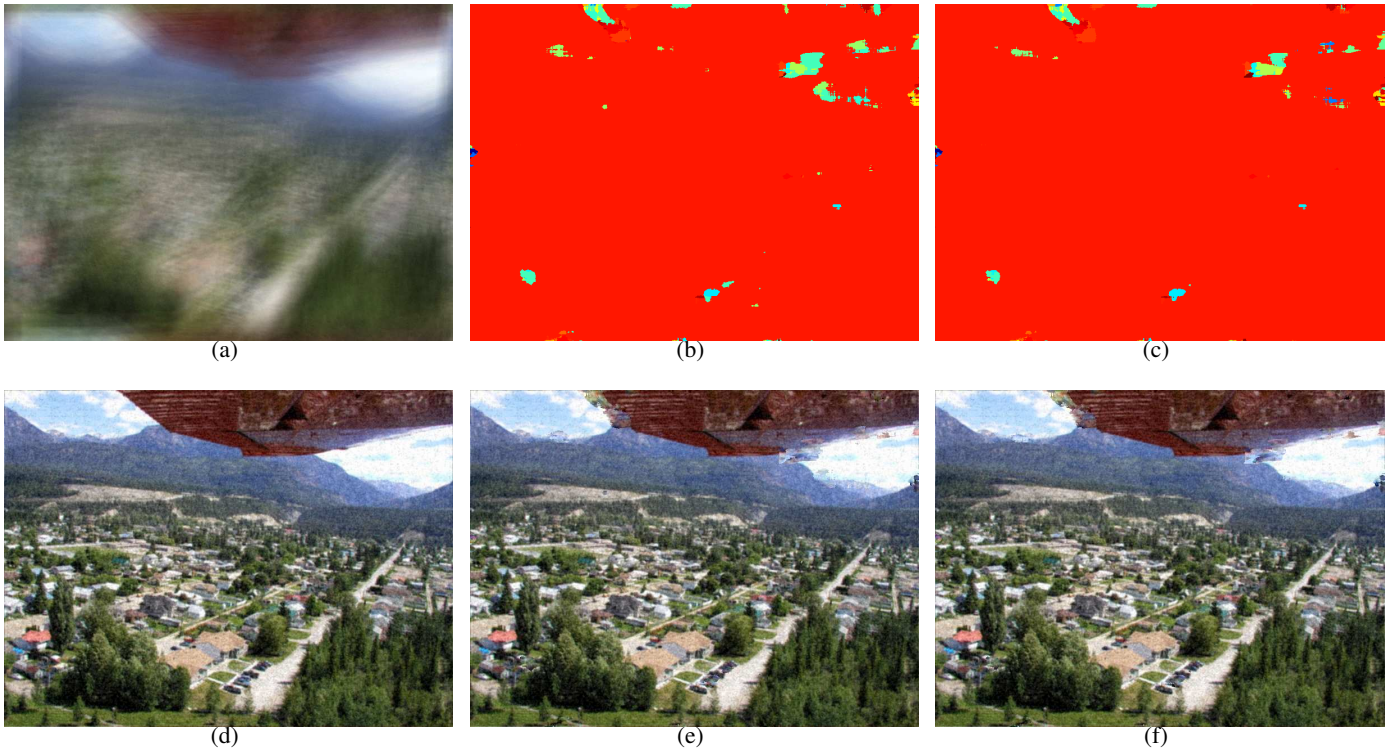


Fig. 17. (a) Input blurry image taken with **a random camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

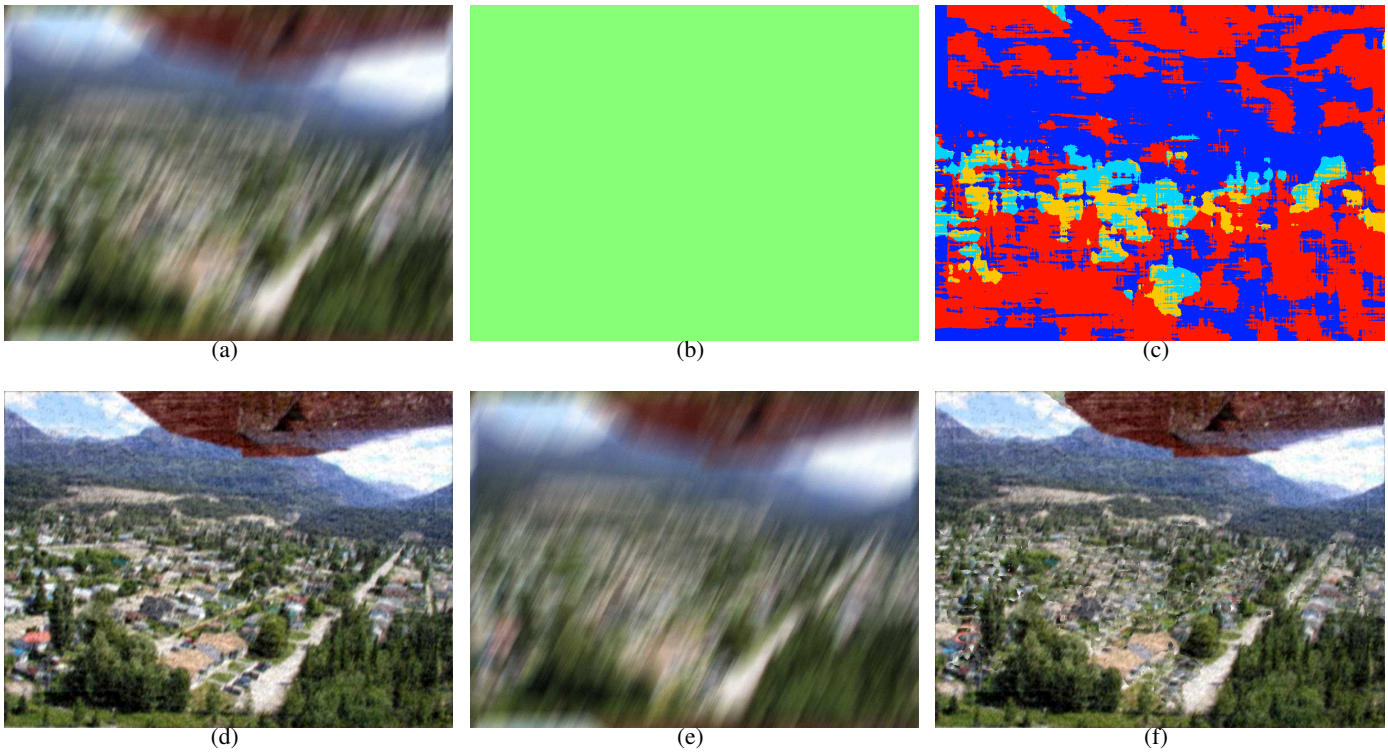


Fig. 18. (a) Input blurry image taken with **a static camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

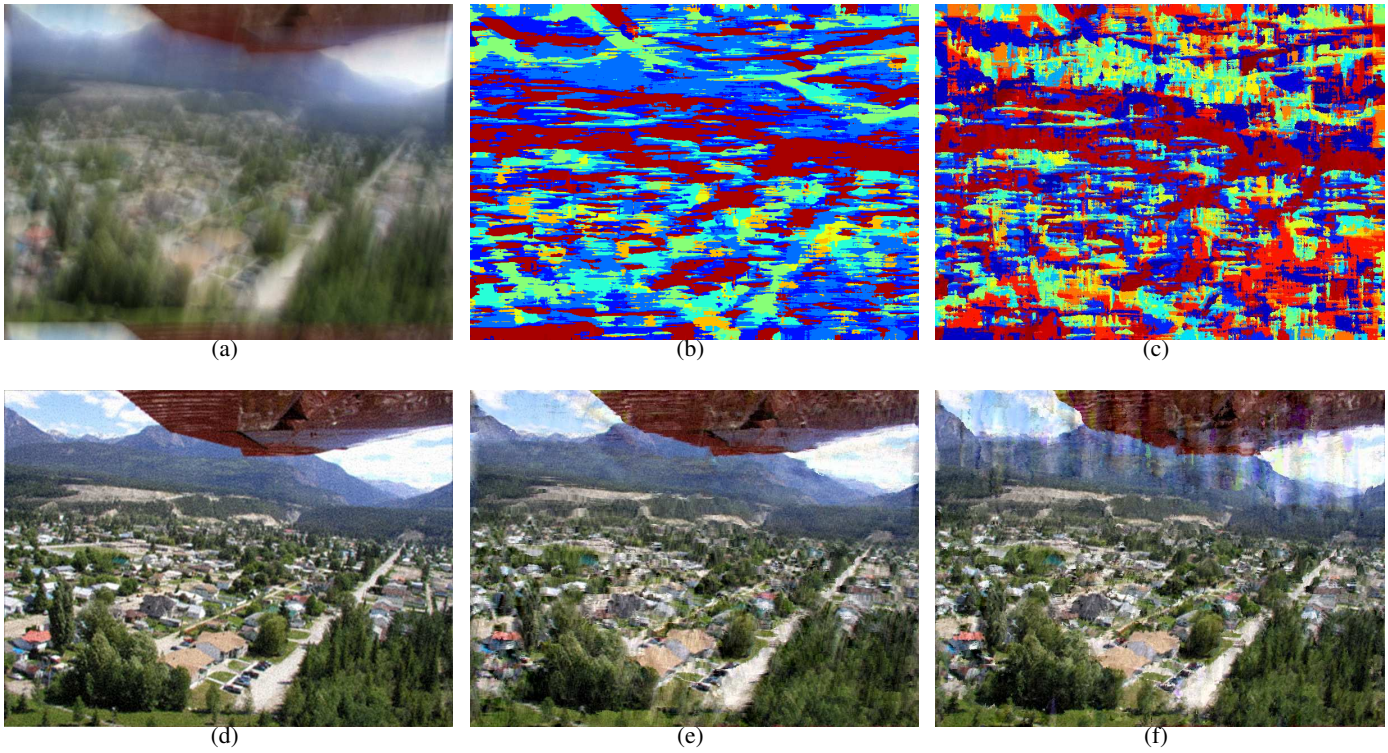
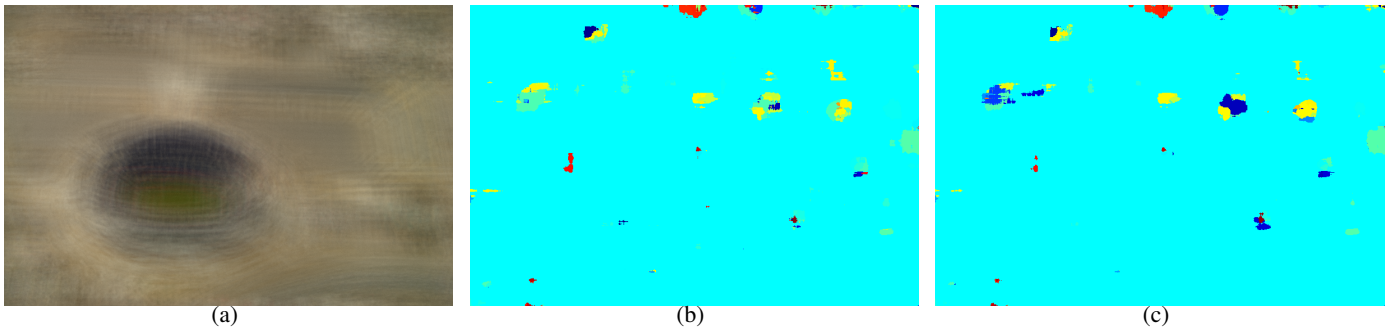


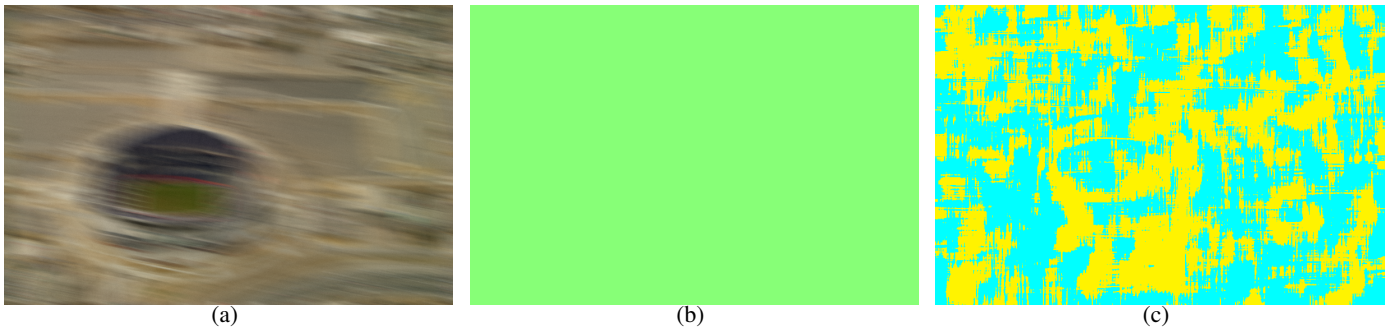
Fig. 19. (a) Input blurry image taken with **a vertical parabolic camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



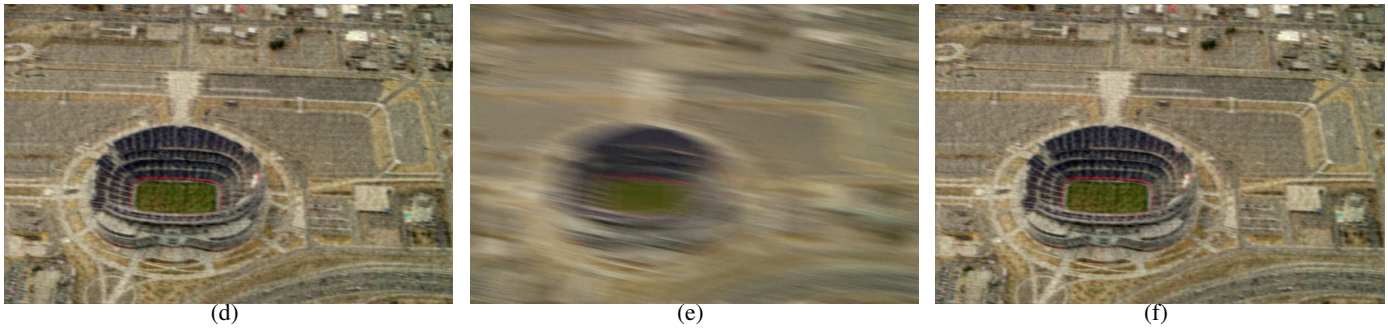
(a) (b) (c)

(d) (e) (f)

Fig. 20. (a) Input blurry image taken with **a random camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

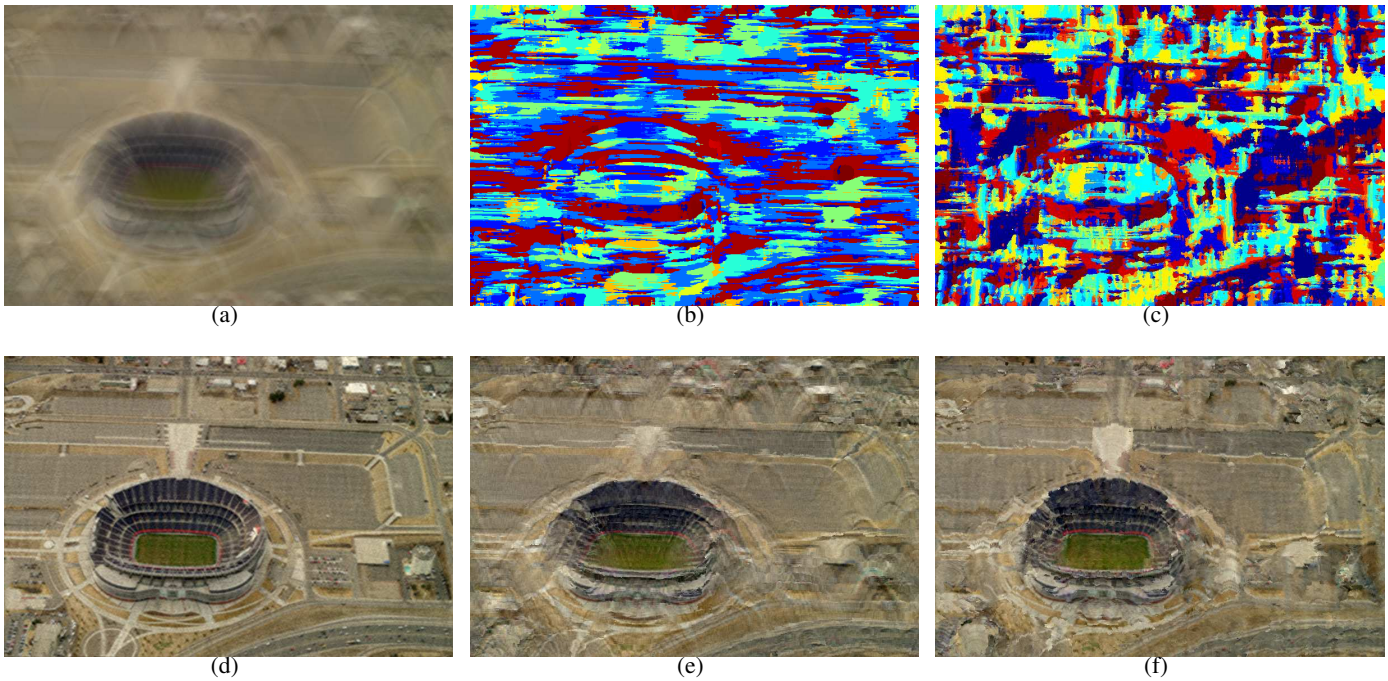


(a) (b) (c)



(d) (e) (f)

Fig. 21. (a) Input blurry image taken with **a static camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

Fig. 22. (a) Input blurry image taken with **a vertical parabolic camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

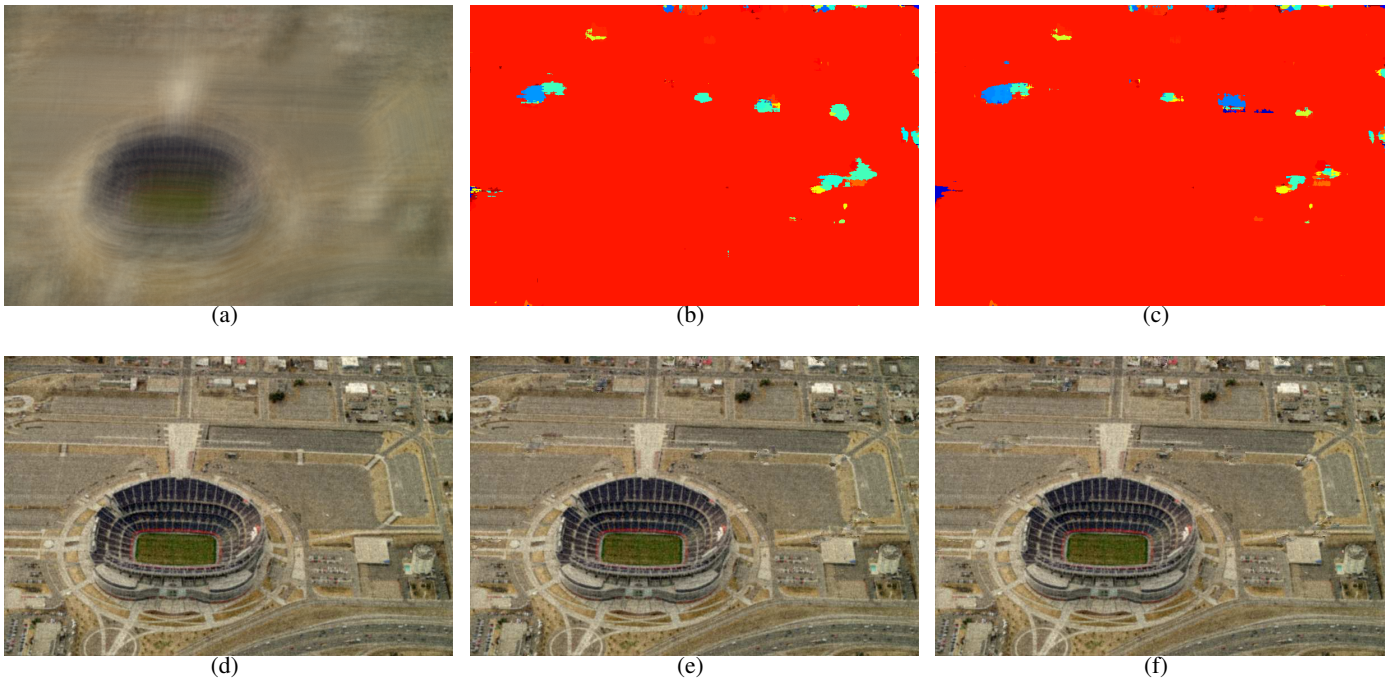


Fig. 23. (a) Input blurry image taken with **a random camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

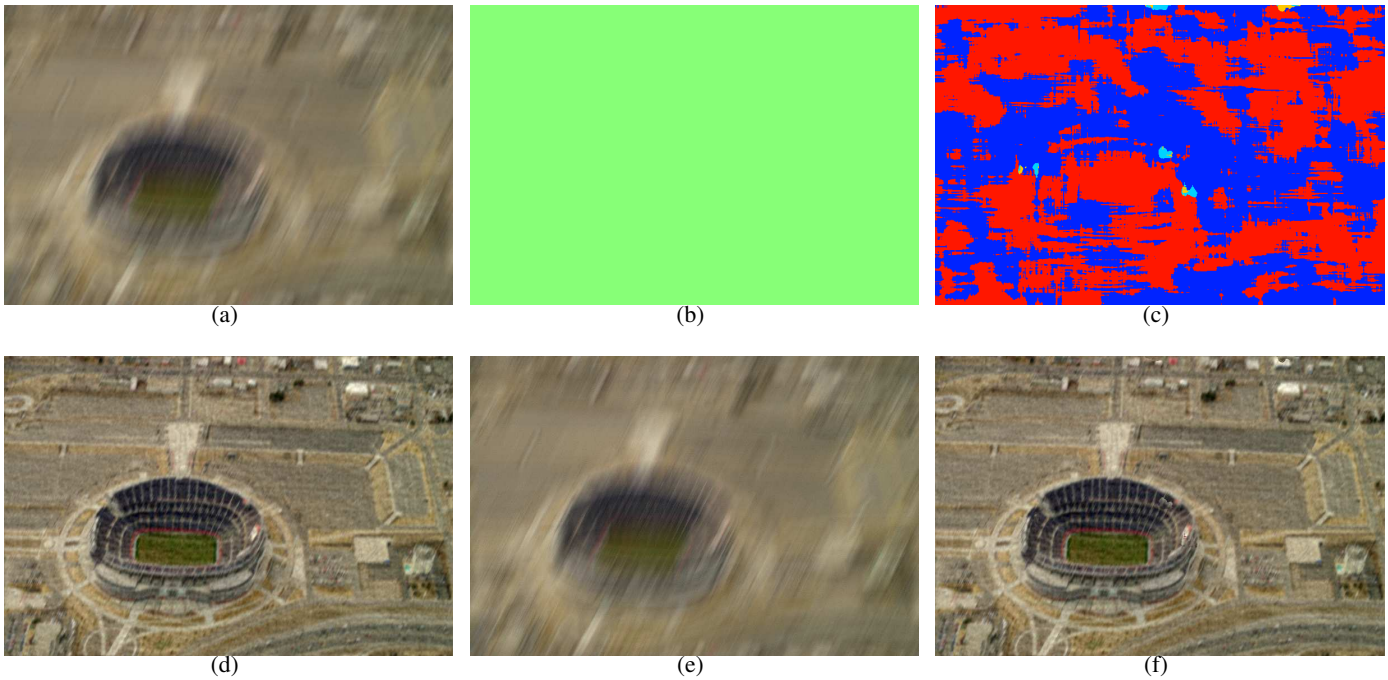
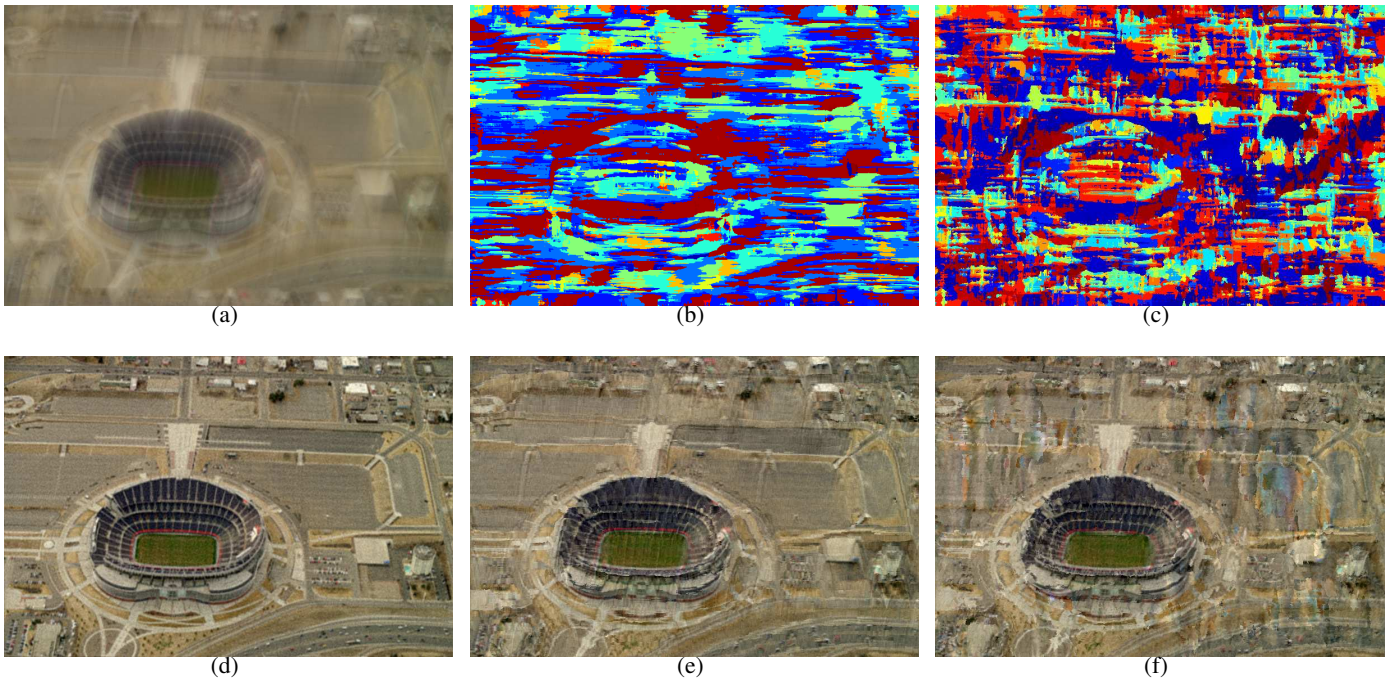


Fig. 24. (a) Input blurry image taken with **a static camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

Fig. 25. (a) Input blurry image taken with **a vertical parabolic camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

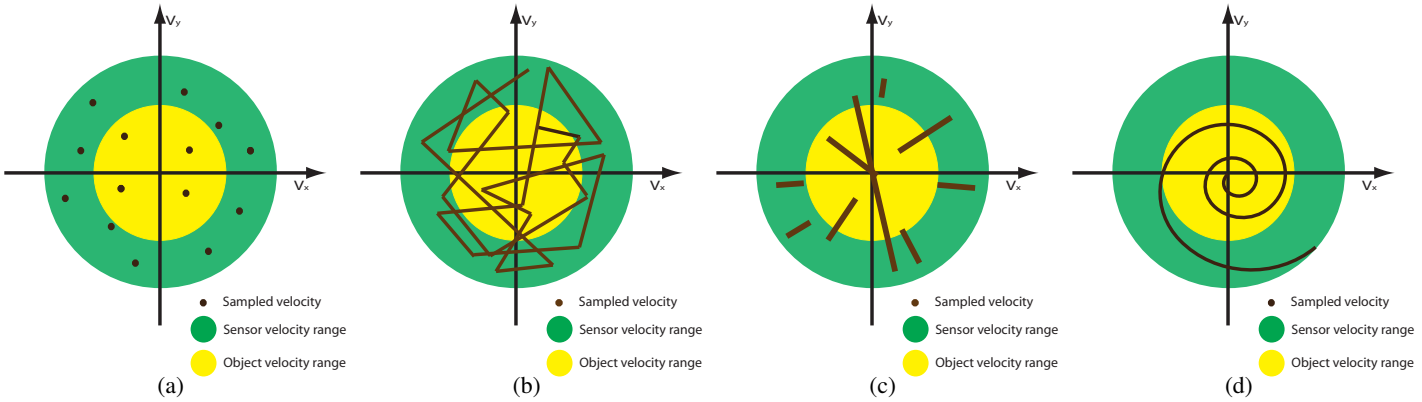


Fig. 26. Velocities covered by the proposed cameras. (a) **The random dot camera** At the start of each segment, the camera randomly chooses a point in the velocity space within the green circle, and moves with that velocity. (b) **The random trace camera** At the start of each segment, the camera randomly chooses a point in the velocity space within the green circle, and moves at a velocity determined by linking velocity chosen in the previous random draw and the current draw. (c) **The random parabolic camera**. At the start of each segment, the camera randomly chooses a velocity within the green cover, and randomly chooses another velocity along the line through the center. The camera moves at a velocity determined by linking those two velocity points. (d) **The spiral camera**. The camera traces the velocity space in a spiral fashion.

IV. Practical camera design

Until now we have evaluated the performance of a rather unrealistic camera that can "hop" - with zero time loss - to generate disconnected camera traces. This is equivalent to arguing that we allow the camera to have a flutter shutter on top of a random "Brownian" motion, and allow more time budget to make up for the time lost during the closed shutter. In this section, we consider more realistic camera designs by assuming that such "hopping" is not allowed, and the camera trace should be continuous. Although this violates the assumption in Eq.(11) that each trace segments are statistically uncorrelated, we show that the performance degradation from violating such an assumption is minimal.

We consider four new cameras, three of which are random in nature: a random dot camera, a random trace camera, and a random parabolic camera, and one of which is deterministic: spiral camera. These names are attributed to the way they cover the velocity space.

A. Camera definitions

Cameras we analyze differ in the way they cover the velocity space. Ideally we want to move the camera in such a way that we will cover the entire velocity space equally. Given a finite time budget and physical constraints, we choose to cover the velocity space with only a small number of velocity samples. Now, the camera design problem becomes how to sample the velocity space dense enough such that $\forall v_x, v_y$, there exists $\hat{v_x}, \hat{v_y}$ sampled by the camera within a ball size of ϵ centered at v_x, v_y. If ϵ is small, the reconstruction error will be small as well.

As in the previous section, we break the integration period into N_s segments, and move with a randomly chosen velocity within each time segments. The random dot camera moves with a constant velocity within each time segment (Figure 26(a)). Unlike the random camera considered in Sec. III, every segment starts at the location the previous segment ended. The random trace camera moves at a speed determined by connecting two random points in the velocity space during each time segment (Figure 26(b)). The random parabolic camera moves at a speed determined by choosing a random point in the velocity space and extending a line through the center during each time segment (Figure 26(c)). The spiral camera moves at a speed determined by covering the velocity space with an Archimedes spiral (Figure 26(d) - note that this spiral is NOT an Archimedes spiral). An Archimedes spiral can be parametrized by $(x, y) = (tcos(t), tsin(t))$.

We choose an Archimedes spiral because the distance between two consecutive layers is always fixed, resulting in small maximum ϵ at all velocities. Unfortunately, as t increases, the camera spends "less" time at each velocity since it has to encircle roughly $2\pi t$ of circumference within 2π of time - the period of the cosine and sine. In other words, the spiral spends more time on small velocities and less on large velocities.

Now, for simulation, instead of linearly changing t, we can non-linearly change t such that the length of the spiral stays the same within each non-linearly sampled time interval. This is quite hard, however, because the length of the spiral is a nonlinear function of t:

$$s(t) = \frac{1}{2}\left\{t\sqrt{1+t^2} + ln\left(t + \sqrt{1+t^2}\right)\right\} \tag{17}$$

What we want is $s(t+\Delta t) - s(t) = const$, and this is hard to solve analytically. Rather than exactly solving for Δt in $s(t+\Delta t) - s(t) = const$, we Taylor-expand $s(t)$ and get

$$\begin{aligned} \Delta s(t) &\approx \Delta t + \frac{1}{6}\left((t+\Delta t)^3 - t^3\right) \\ &= \frac{1}{6}\left((\Delta t)^3 + 3t(\Delta t)^2 + 3t^2\Delta t + 6\Delta t\right) \end{aligned} \tag{18}$$

If $\Delta t \to 0$, we can ignore $(\Delta t)^3$:

$$\Delta t \approx \frac{\sqrt{(3t^2+6)^2 + 18\Delta s t} - (3t^2+6)}{6t} \tag{19}$$

where Δs is the constant length of the spiral. We can now trace the velocity space roughly equally at all velocities using this t parametrization. Since that's still an approximation, we will see some variation in the weighted reconstruction error when using the spiral camera.

We can immediately analyze the performance of these cameras: the random trace camera will do slightly better than the random dot camera, especially when N_s is small, because it covers more velocities; the weighted reconstruction error in the random parabolic camera will favor small velocities since each velocity segments are directed towards the center, covering more "small" motions (i.e. ϵ for small v_x, v_y will be smaller than that of large v_x, v_y); the spiral camera expects to do well in terms of weighted reconstruction error, but the PSF estimation performance is expected to be poor since blur kernels can be expected to be quite similar as the object motion changes. We address these issues experimentally in the next few sections. We fix $S_{ms} = 2 \times S_{mo}$, $N_s = 50$ in all experiments shown in this section.

B. The PSF and the spectrum

In this section, we show some PSFs and their log spectrums for the random dot camera, the random trace camera, the random parabolic camera, and the spiral camera.

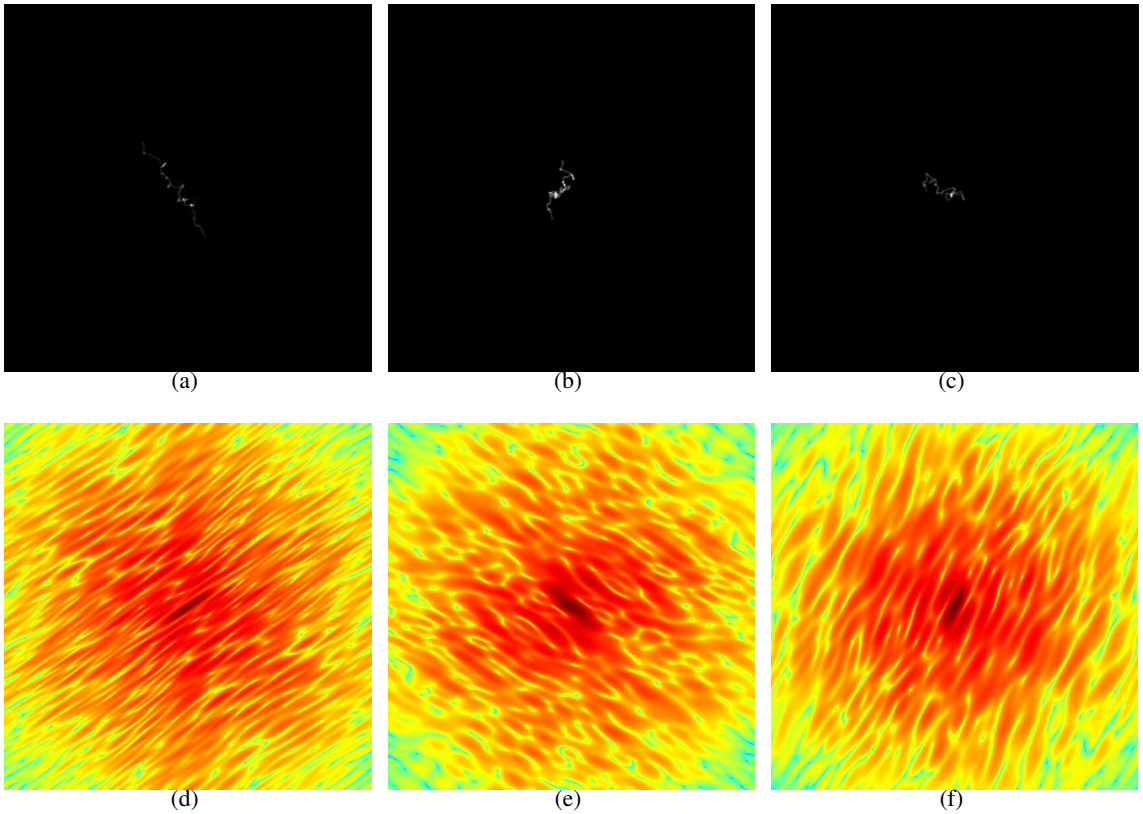


Fig. 27. Top row: the PSF of the **random dot** camera with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively. Bottom row: the log spectrum of the random dot camera with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively.

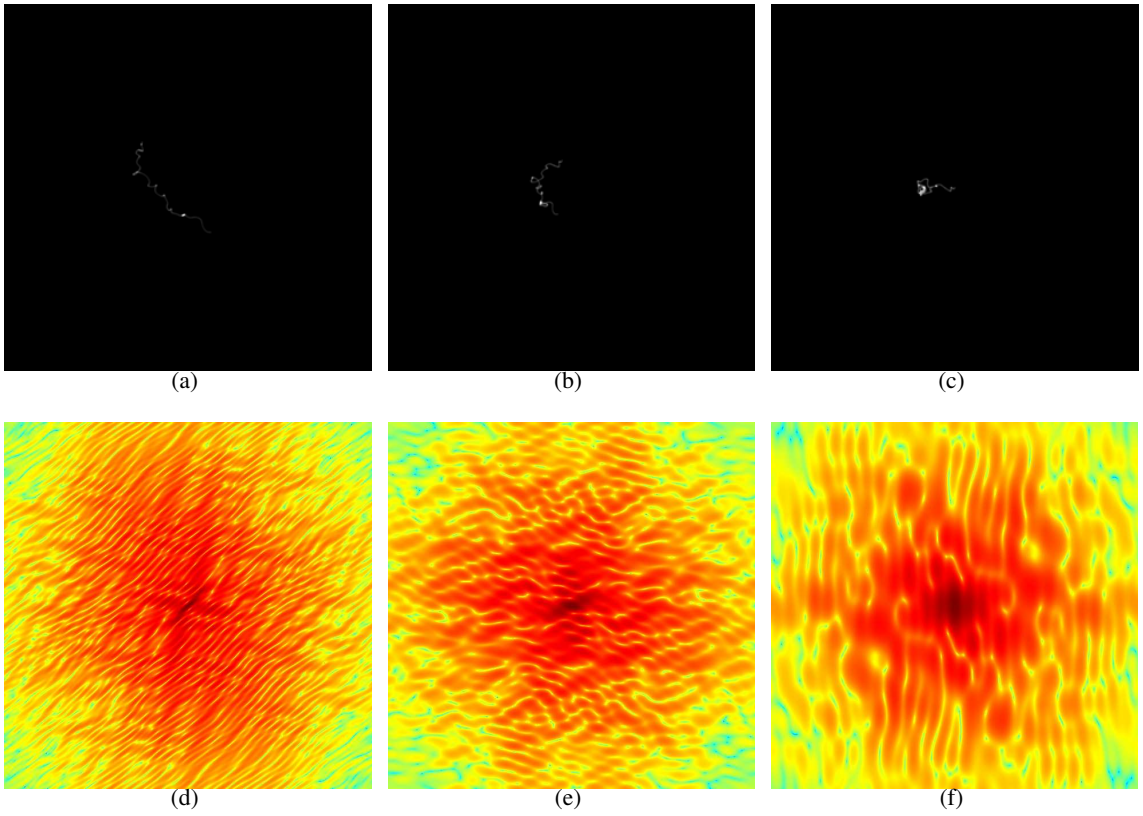


Fig. 28. Top row: the PSF of the **random trace camera** with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively. Bottom row: the log spectrum of the random trace camera with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively.

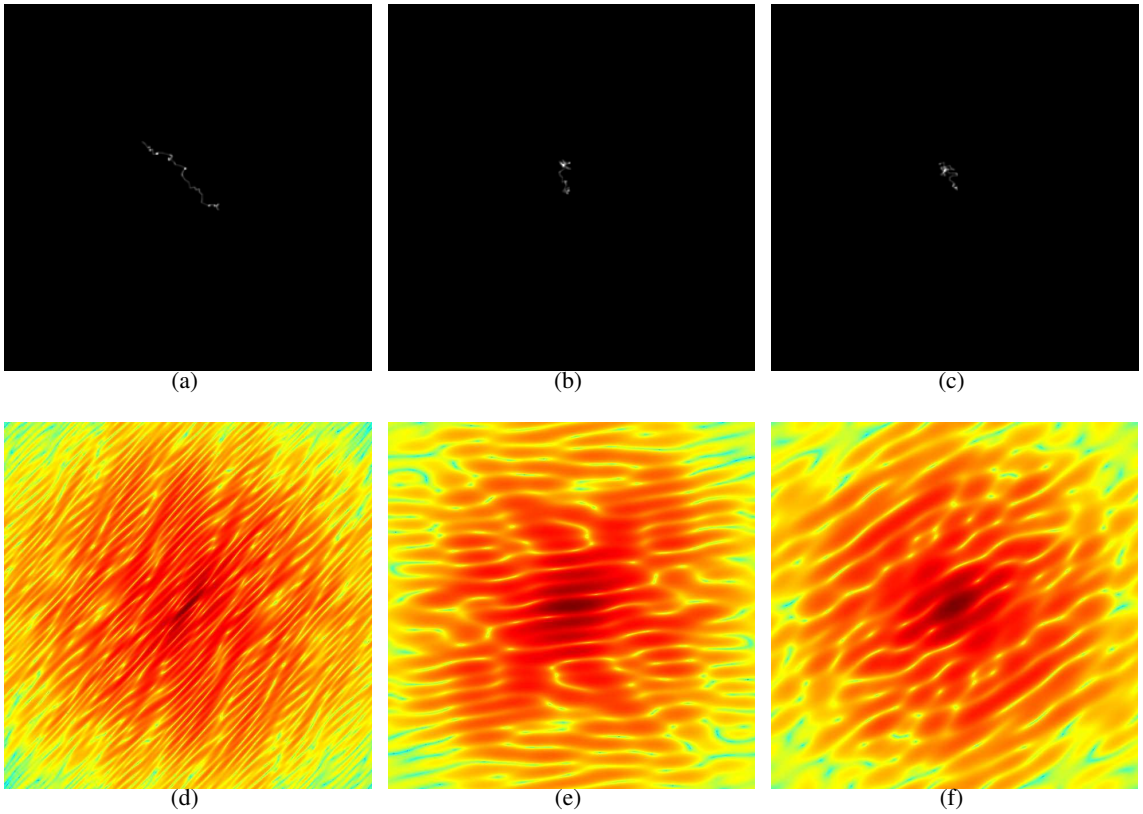


Fig. 29. Top row: the PSF of the **random parabolic camera** with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively. Bottom row: the log spectrum of the random trace camera with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively.

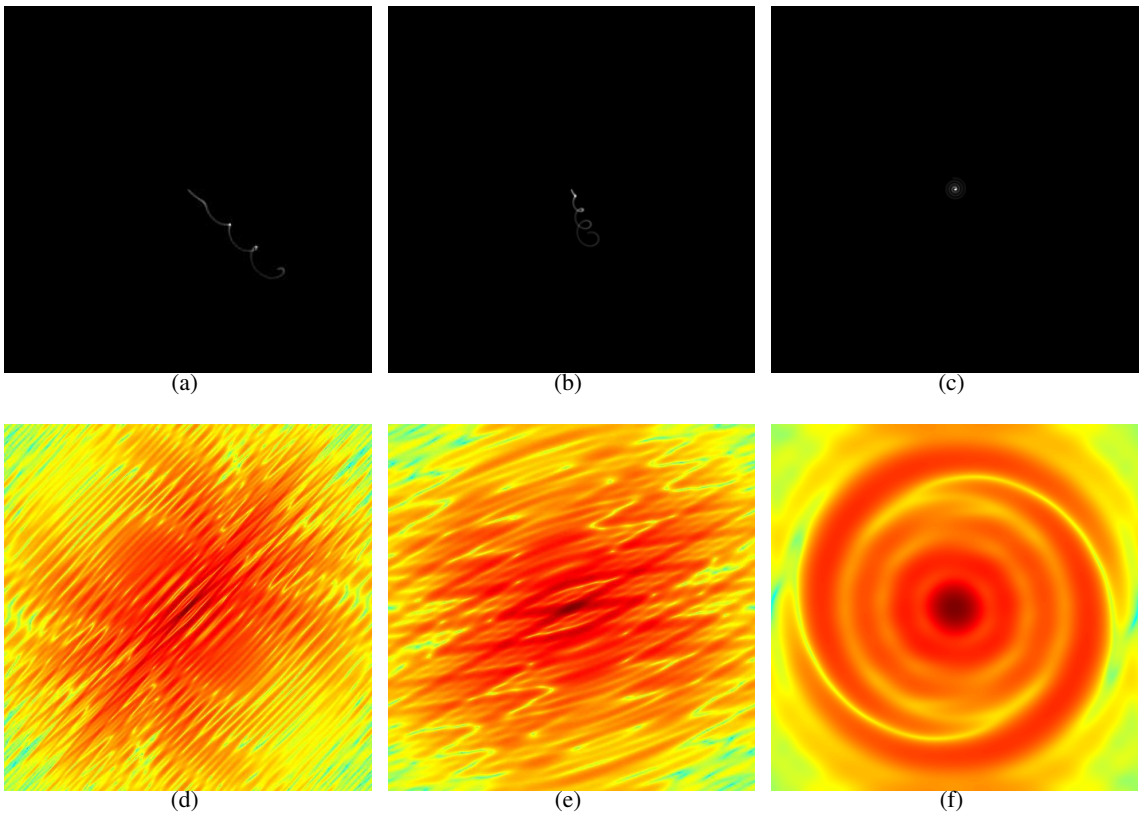
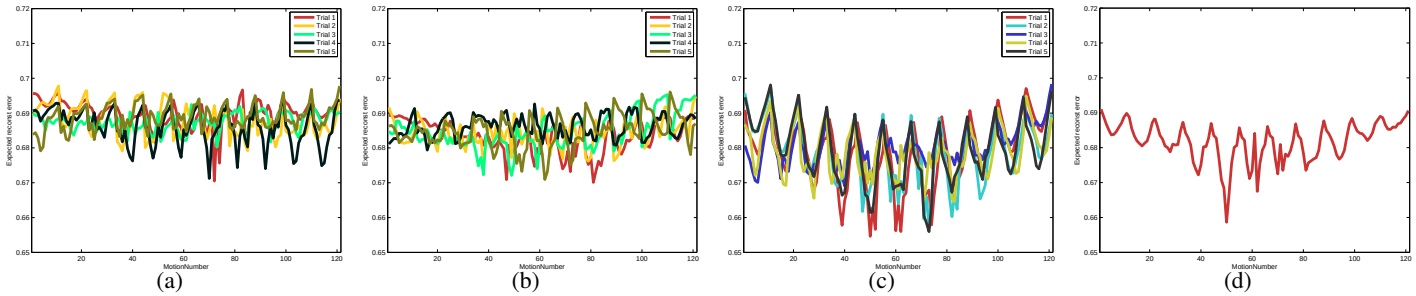
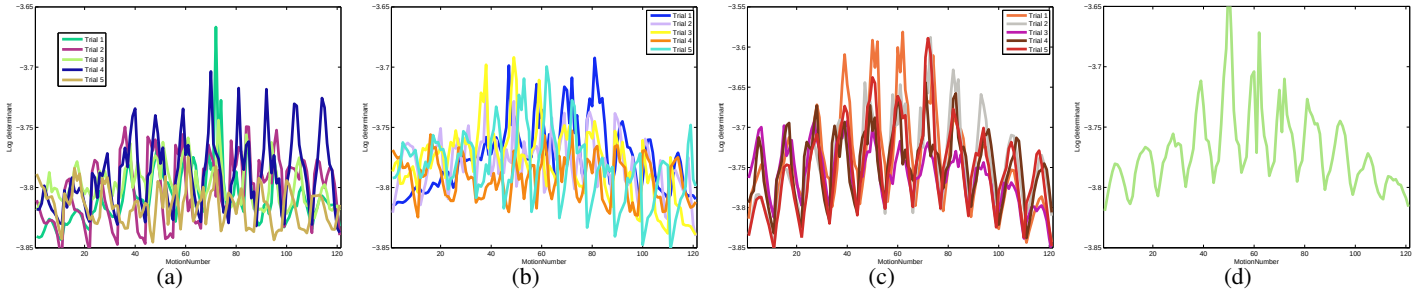


Fig. 30. Top row: the PSF of the **spiral camera** with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively. Bottom row: the log spectrum of the spiral camera with object motion 1 (very fast diagonal), 26 (moderate diagonal), 61 (static), respectively.



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

Fig. 31. (a) Weighted reconstruction errors for a **random dot camera**. (b) Weighted reconstruction errors for a **random trace camera**. (c) Weighted reconstruction errors for a **random parabolic camera**. (d) Weighted reconstruction errors for a **spiral camera**.



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

Fig. 32. (a) Log determinants for a **random dot camera**. (b) Log determinants for a **random trace camera**. (c) Log determinants for a **random parabolic camera**. (d) Log determinants for a **spiral camera**.

C. Weighted reconstruction error and the log determinants

We show the weighted reconstruction error and the log determinants for the proposed cameras in Figure 31 and Figure 32, respectively. The weighted reconstruction errors for a random dot camera (Figure 31) has more variations than the disconnected counterpart, but on average the weighted reconstruction error is a bit better than that of the disconnected counterpart. The correlation term in Eq.(15) should be helping the weighted reconstruction error. The maximum weighted reconstruction error is smaller than the maximum error of the parabolic camera.

The variation in the weighted reconstruction is less with a random trace camera, and its performance is also better than that of a disconnected random dot camera. The reduced variation can be attributed to the fact that we cover more velocities within the integration time. As expected, the random parabolic camera favors small motions and achieves a very small error near (in 2D distance in the motion matrix) motion 61. However, the variation in the weighted reconstruction error is much larger than the random dot camera and the random trace camera. The weighted reconstruction error for the spiral camera is somewhat large as well. The approximation may not be accurate, and another form of spiral may seem to be tried. However, the spiral camera doesn't do well in terms of PSF estimation either, as we show later, and thus it seems justifiable to reject the spiral camera anyhow.

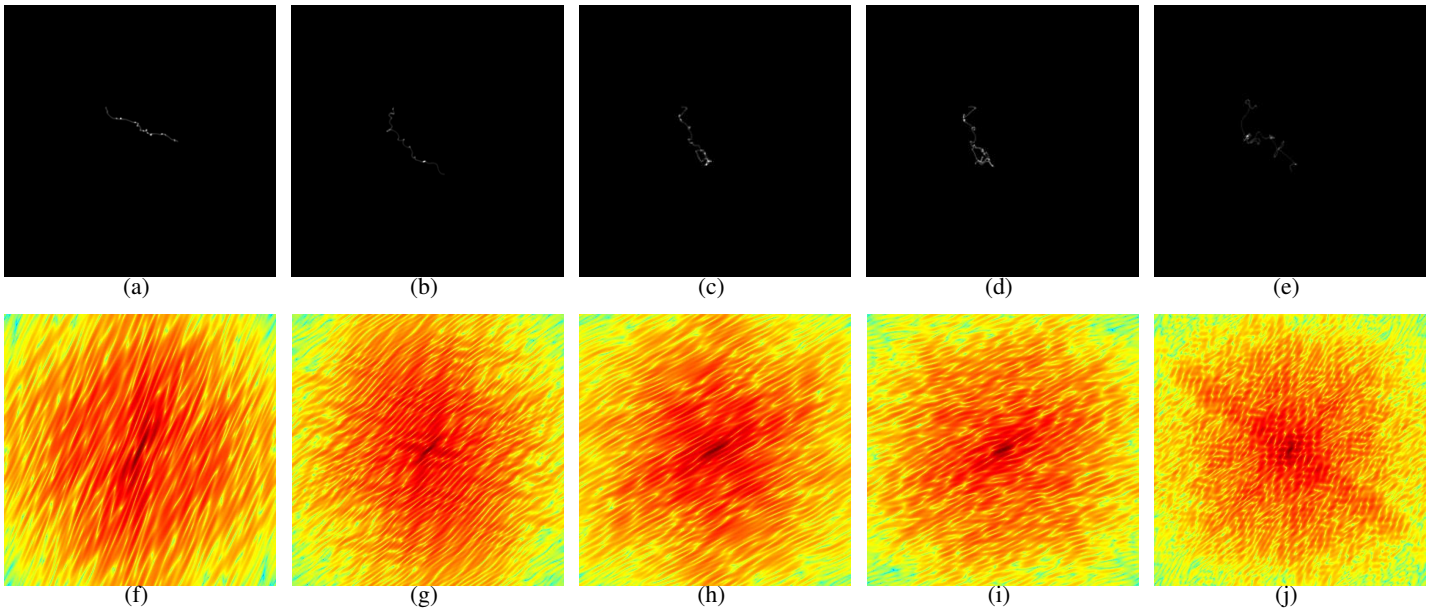


Fig. 33. Top row: the PSF of **the random camera** for motion 1 (very fast diagonal) when (a) $S_{ms} = 1.5 \times S_{mo}$, (b) $S_{ms} = 2 \times S_{mo}$, (c) $S_{ms} = 3 \times S_{mo}$, (d) $S_{ms} = 4 \times S_{mo}$, (3) $S_{ms} = 5 \times S_{mo}$. Bottom row: the log spectrum of the kernels in the top row.

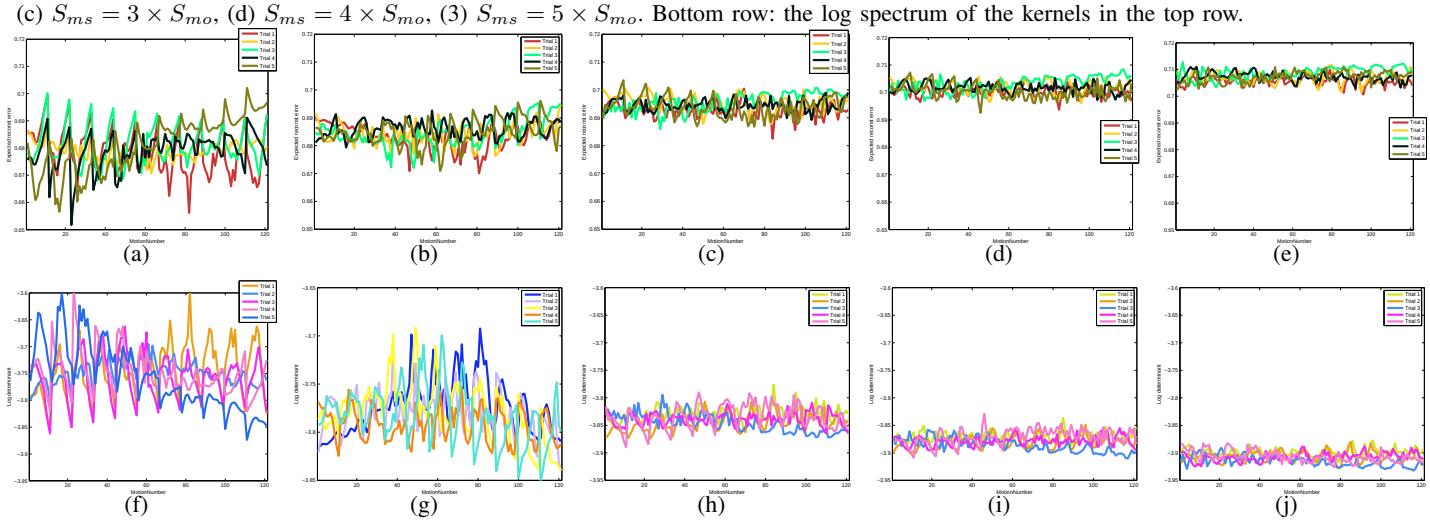


Fig. 34. Top row: The weighted reconstruction error of **the random camera** when (a) $S_{ms} = 1.5 \times S_{mo}$, (b) $S_{ms} = 2 \times S_{mo}$, (c) $S_{ms} = 3 \times S_{mo}$, (d) $S_{ms} = 4 \times S_{mo}$, (e) $S_{ms} = 5 \times S_{mo}$. Bottom row: the log determinant of the random camera when (f) $S_{ms} = 1.5 \times S_{mo}$, (g) $S_{ms} = 2 \times S_{mo}$, (h) $S_{ms} = 3 \times S_{mo}$, (i) $S_{ms} = 4 \times S_{mo}$, (j) $S_{ms} = 5 \times S_{mo}$.

D. Varying S_{ms} and N_s in the random trace camera

In the previous section, we fixed S_{ms} and N_s in computing the weighted reconstruction error and the log determinant. In this section, we want to analyze how S_{ms} and N_s affect the performance of a random trace camera. As with the random camera with disconnected paths, we can anticipate that as we increase S_{ms}, the "average" (over object motions) weighted reconstruction error will increase (since the spectral power is shared among larger range of velocities), but the variation of the weighted reconstruction error among different object motions will decrease, especially for fast object motions. As we increase N_s, the variation of the weighted reconstruction error among different object motions will decrease, and also the "average" (over object motions) weighted reconstruction error will decrease. However, we cannot indefinitely increase N_s due to lens blur, finite sensor resolution, physical implementation constraints etc...

1) Sweeping S_{ms}: To verify the performance dependence on S_{ms}, we vary the value of S_{ms} and plot the weighted reconstruction error and the log determinant. Again, for each S_{ms} value we sampled 5 different camera movements. Values of S_{ms} considered are $1.5 \times S_{mo}$, $2 \times S_{mo}$, $3 \times S_{mo}$, and $4 \times S_{mo}$ and $5 \times S_{mo}$. We fix N_s to 50 in this section. The noise level was 0.01 in this experiment. Figure 33 shows some random PSF's and their respective log

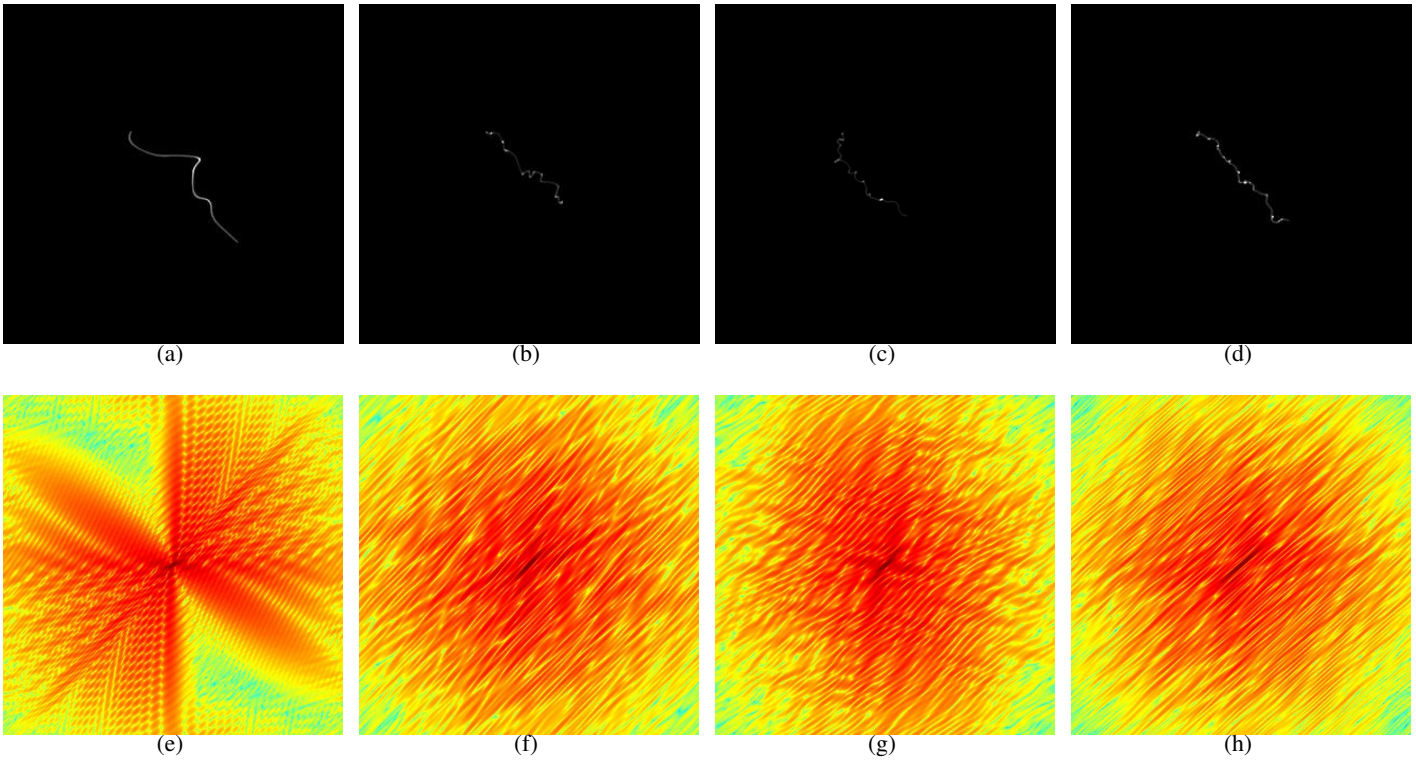


Fig. 35. Top row: the PSF of **the random trace camera** for motion 1 (very fast diagonal) when (a) $N_s = 10$, (b) $N_s = 30$, (c) $N_s = 50$, (d) $N_s = 70$. Bottom row: the log spectrum of the kernels in the top row.

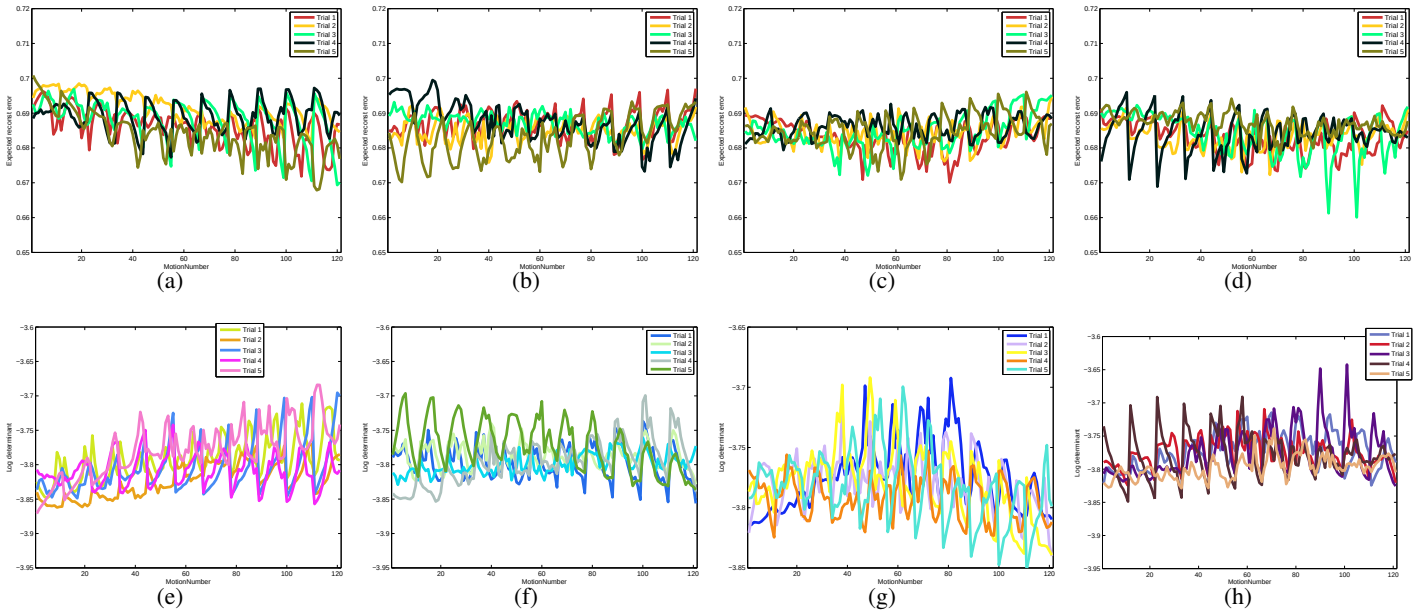


Fig. 36. Top row: the weighted reconstruction error of **the random trace camera** when (a) $N_s = 10$, (b) $N_s = 30$, (c) $N_s = 50$, (d) $N_s = 70$. Bottom row: the log determinant of the random camera when when (e) $N_s = 10$, (f) $N_s = 30$, (g) $N_s = 50$, (h) $N_s = 70$.

spectrums for a fast diagonal object movement. The log spectrums all fall off proportional to the distance from the origin, and have distinguishable zero structures.

It's hard to quantify the characteristic just by looking at the log spectrums. So we plot the weighted reconstruction error and the log determinants for the random trace camera in Figure 34. One evident feature in the weighted reconstruction error is that as we increase S_{ms}, the average weighted reconstruction error increases as well. Unlike the random dot camera with disconnected paths, the random trace camera is more severely affected by S_{ms}: at

TABLE I
KERNEL SELECTION ACCURACY AT $\eta = 0.01$

Camera type	% of correct selections	
	Est. Err. based	Log Like. based
Random dot camera	121 / 121	121 / 121
Random trace camera	121 / 121	121 / 121
Random parabolic camera	113 / 121	121 / 121
Spiral camera	68 / 121	121 / 121

$S_{ms} \approx 1.5 \times S_{mo}$, the variation in the weighted reconstruction error is quite large, although $S_{ms} = 1.5 \times S_{mo}$ was large enough for the random dot camera with disconnected paths. One reason for this observation is that it's harder to overcome the elongated shape of the kernel generated by the object motion when the camera traces are connected. The reason for increasing S_{ms} is that the kernel support (or the randomness in the kernel) does not get affected by the object movement. If the camera trace is all connected, it's harder to make the kernel to be unbiased to the object movement, as you can see from Figure 33. From the graphs, we see that the variation of the reweighted reconstruction is tolerable once $S_{ms} \approx 2 \times S_{mo}$.

2) Sweeping N_s: To verify the performance dependence on N_s, we vary the value of N_s and plot the weighted reconstruction error and the log determinant. Again, for each N_s value we sampled 5 different camera movements. Values of N_s considered are 10, 30, 50, and 70. We fix S_{ms} to $2 \times S_{mo}$ in this section. The noise level was 0.01 in this experiment. Figure 35 shows some random PSF's and their respective log spectrums for a fast diagonal object movement. When N_s is small, we can observe the elongated directional structure in the spectrum, and the zero structures are quite large. As we increase N_s, the elongated structure in the spectrum disappears, and the zero structures become much finer as well.

The weighted reconstruction error and the log determinant plots as we sweep N_s are shown in Figure 36. As we increase N_s, the variation in the weighted reconstruction error and the log determinants becomes smaller, but the average value of the weighted reconstruction error stays pretty much the same. From these results, we fix $N_s = 50$ and $S_{ms} = 2 \times S_{mo}$ for the subsequent experiments.

E. Identifying the PSF from the input image

In this section, we estimate the object motion (i.e. the PSF) from the input blurry image. Again, we consider both the whole image based and local patch based PSF estimation. 1% noise was added to all blurred images. The whole image based PSF estimation is more stable because there are more observations.

1) PSF estimation using the whole image: The PSF estimation performance of each camera is summarized in Table I. For images blurred with every 121 kernels in the pool, the algorithm chose the PSF - from a pool of 121 different kernels - that the algorithm believes was used in blurring the image. As expected, the estimation error based measure works reasonable well in most cameras, and works perfectly for the random dot camera and the random trace camera. The random parabolic camera does well, but is not perfect even given the whole image. The possible PSF identification error could be from the fact that all velocity segments are directed towards/away from the origin, and that may result in similar zeros for similar object motions. The estimation error based measure doesn't work too well with the spiral camera. This can be intuitively understood by noting that the PSF shape doesn't change much as the object motion varies slightly. Therefore the zeros in the spectrum are likely to lie at similar locations for similar motions, leading to ambiguities. When the log determinant based measure is used, the random parabolic camera, as well as the spiral camera correctly identify all kernels.

We plot the confusion matrices for the estimation error based classification and the log likelihood based classification in Figure 37. The confusion matrix is row-wise normalized. The confusion matrix is 121 x 121, and the j^{th} column of the i^{th} row is the score given to the hypothesis that the image originally blurred with i^{th} kernel is identified as being blurred with j^{th} kernel. Higher score means that the algorithm thinks it's more likely. Ideally we want to have a single dark red diagonal component and the rest dark blue. We can observe from the first row of Figure 37 that the random dot camera and the random trace camera are very stable in estimating the correct kernel from the estimation error based measure. What's interesting is the spiral camera's confusion matrix. The

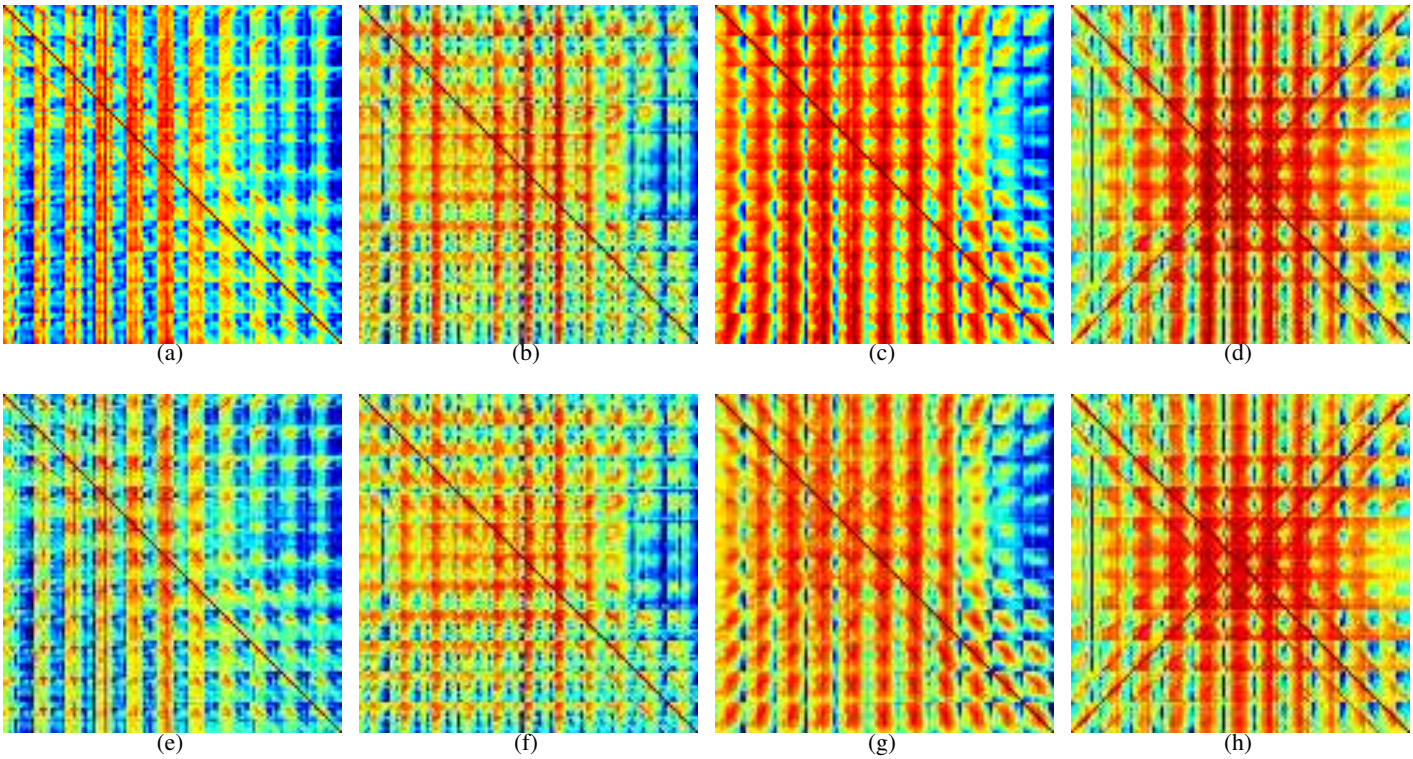


Fig. 37. Red signifies a high value, and blue signifies a low value. **Top row**: Confusion matrices for PSF identification using the estimation error based measure (a) The random dot camera, (b) The random trace camera, (c) The random parabolic camera, (d) The spiral camera. **Bottom row**: Confusion matrices for PSF identification using the log likelihood based measure (e) The random dot camera, (f) The random trace camera, (g) The random parabolic camera, (h) The spiral camera.

spiral camera has a hard time discriminating the correct object motion from the the motion in the opposite direction - note the two diagonal lines in the confusion matrix.

2) PSF estimation from local patches: In this section, we estimate the PSF from the image using local patches. Our test set is two images, each blurred with two different PSFs, and our goal is to correctly identify the PSF that blurred the two images from a pool of 121 blur kernels. For each test case, we test the random dot camera with connected camera movement, the random trace camera, the random parabolic camera, and the spiral camera. The previous section tells us that the random dot camera and the random trace camera will be better in estimating the correct kernel than the other two cameras. The patch size is 41 x 41 in this experiment.

The PSF estimation results are shown in the subsequent pages. One thing to notice is that the PSF estimation performance of these cameras is worse than that of the random dot camera with disconnected camera movement. Using the log determinant does help in correctly identifying the kernels in these new cameras, but the estimation performance is still not comparable to that of the random dot camera with disconnected paths. One way to explain this observation is that by connecting different segments we are introducing regularities, making the kernel not entirely random. We can reduce the regularity of connected paths by varying the length of each segment, but this hasn't been tried yet.

The misclassification usually occurs when there's a change in the texture. We may be able to reduce small islands of errors by using the graph-cut based motion layer smoothing, but such large error around the texture boundaries is hard to reduce. We could increase the size of the neighborhood patch, but this hasn't been tried yet.

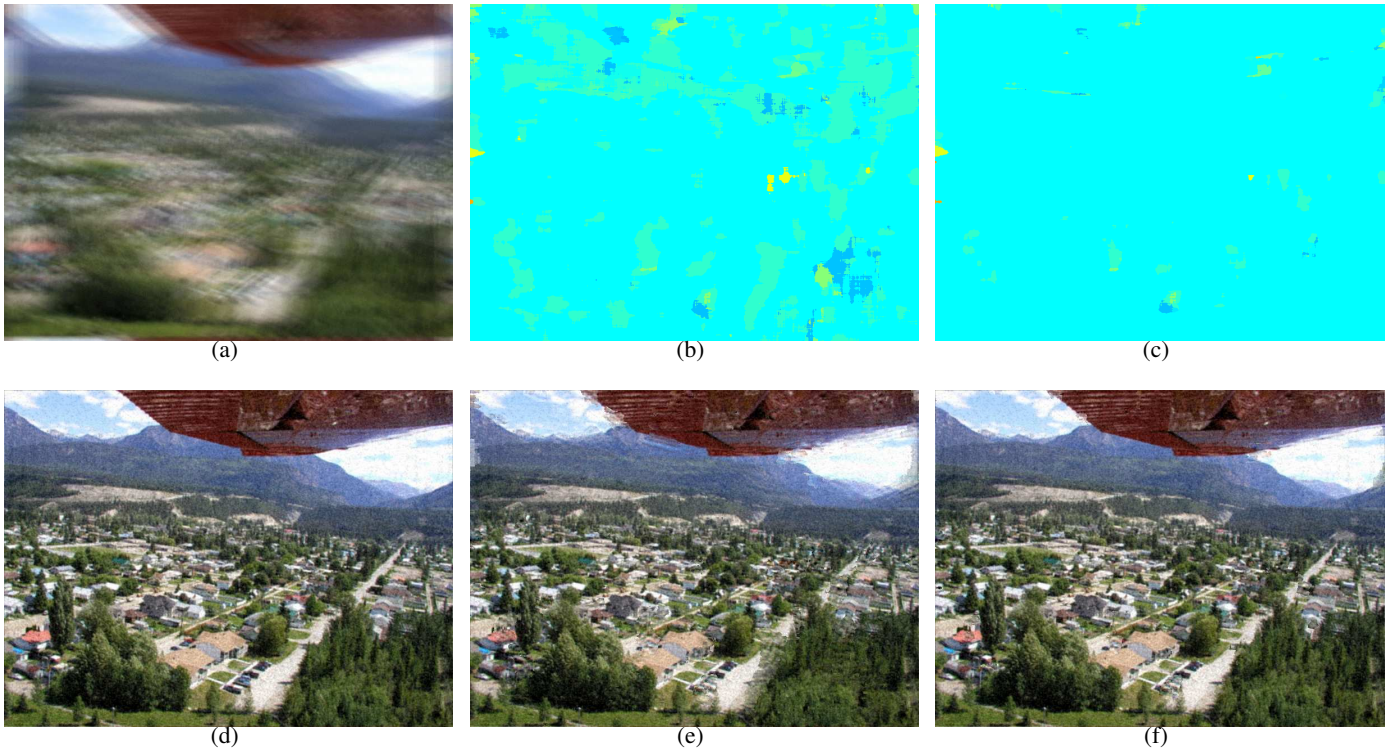


Fig. 38. (a) Input blurry image taken with **a random dot camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

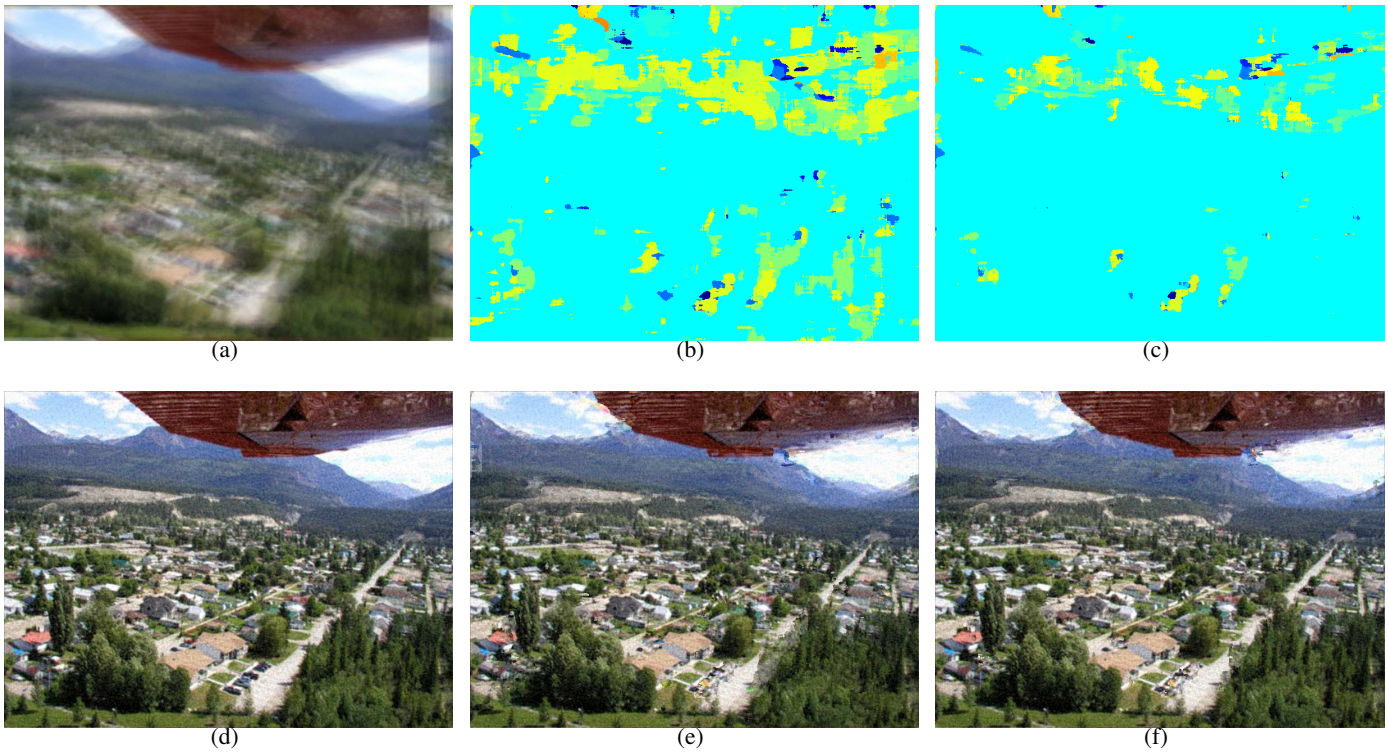


Fig. 39. (a) Input blurry image taken with **a random trace camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

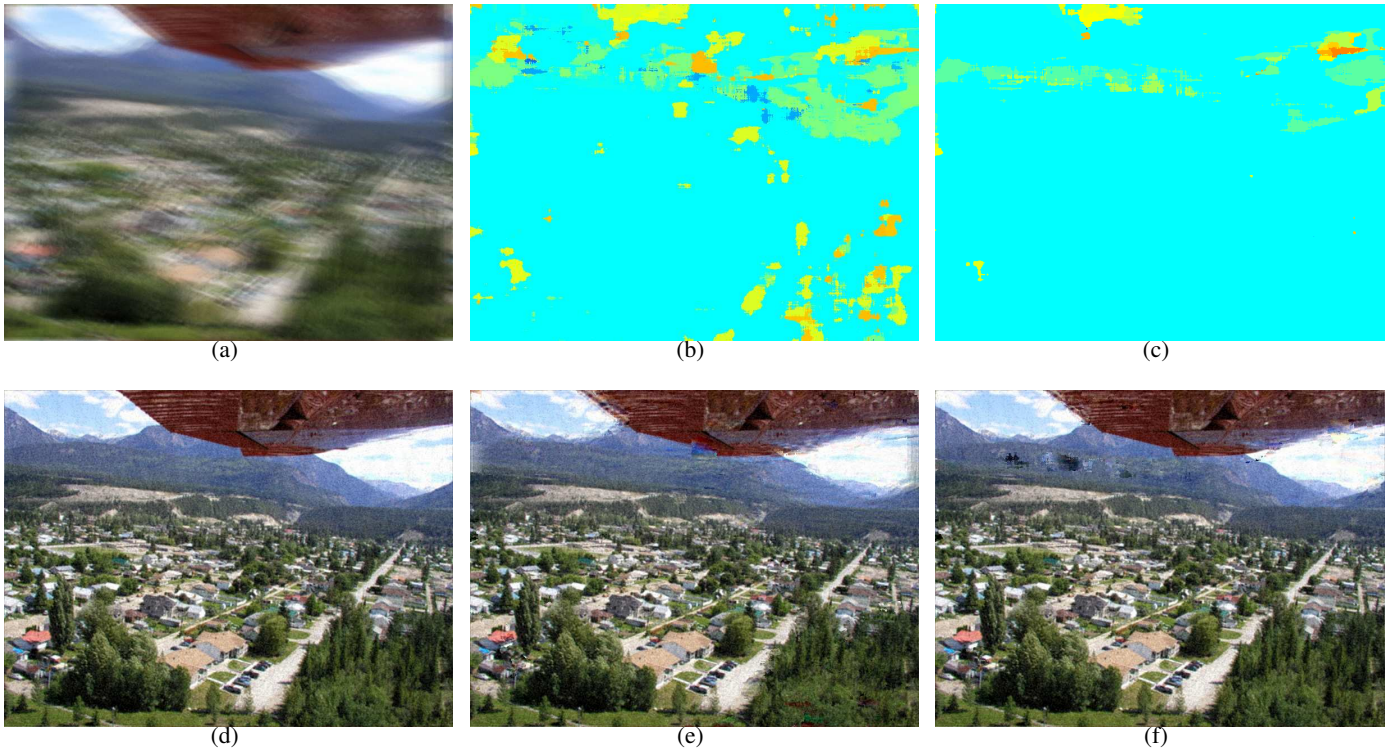


Fig. 40. (a) Input blurry image taken with **a random parabolic camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

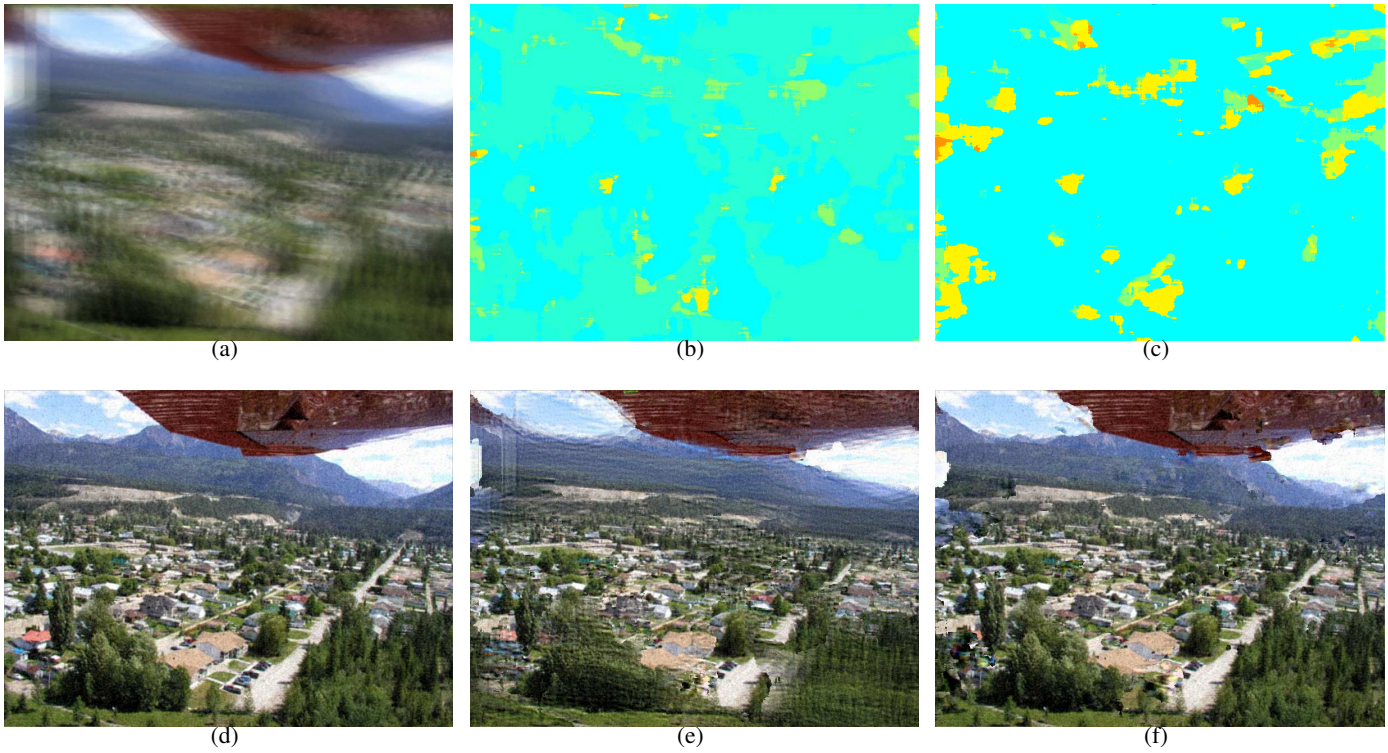


Fig. 41. (a) Input blurry image taken with **a spiral camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

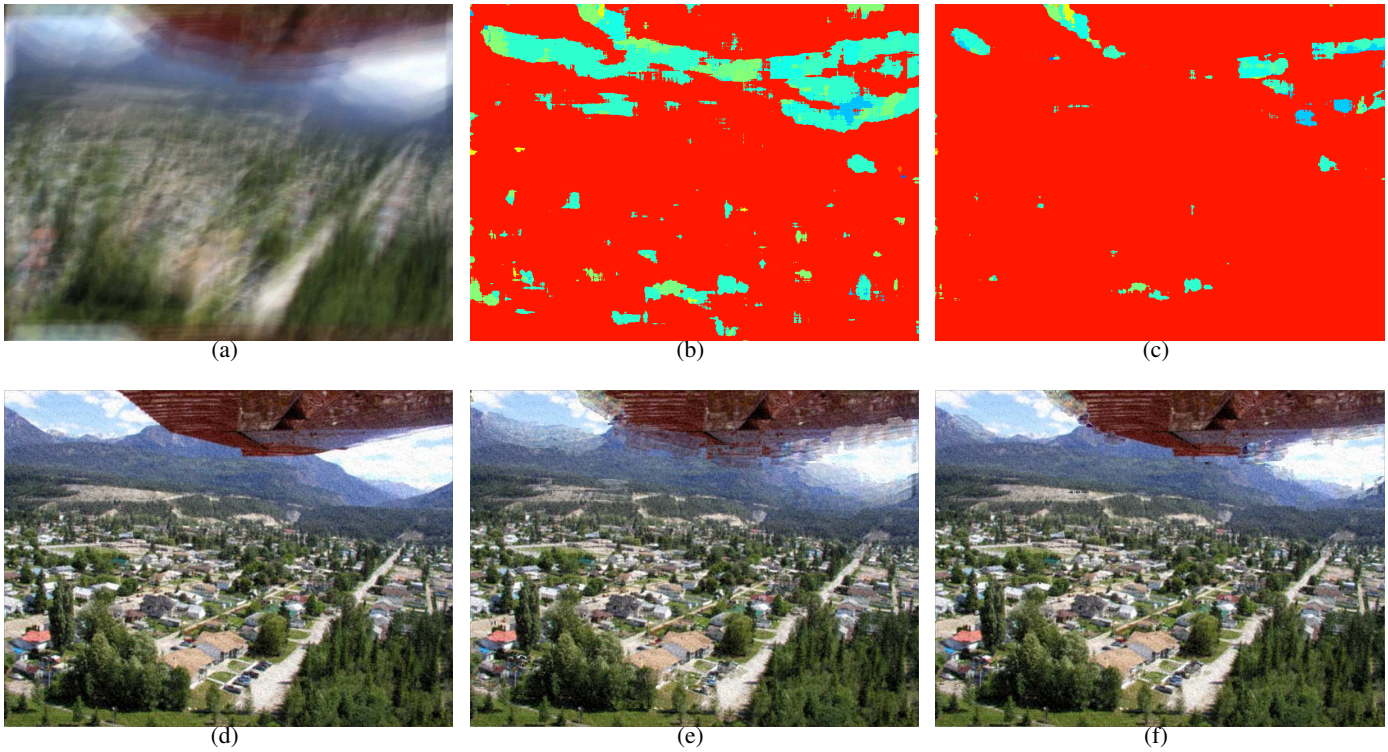


Fig. 42. (a) Input blurry image taken with **a random dot camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

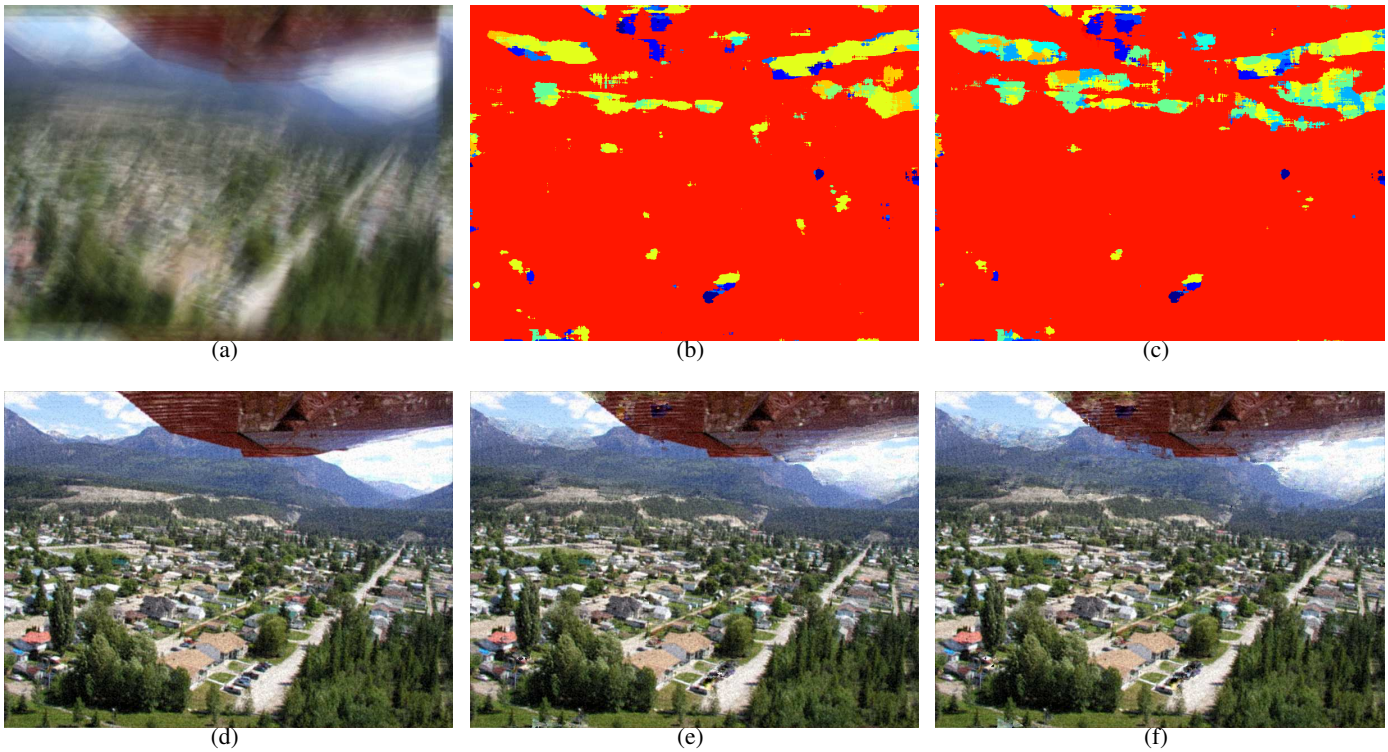


Fig. 43. (a) Input blurry image taken with **a random trace camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

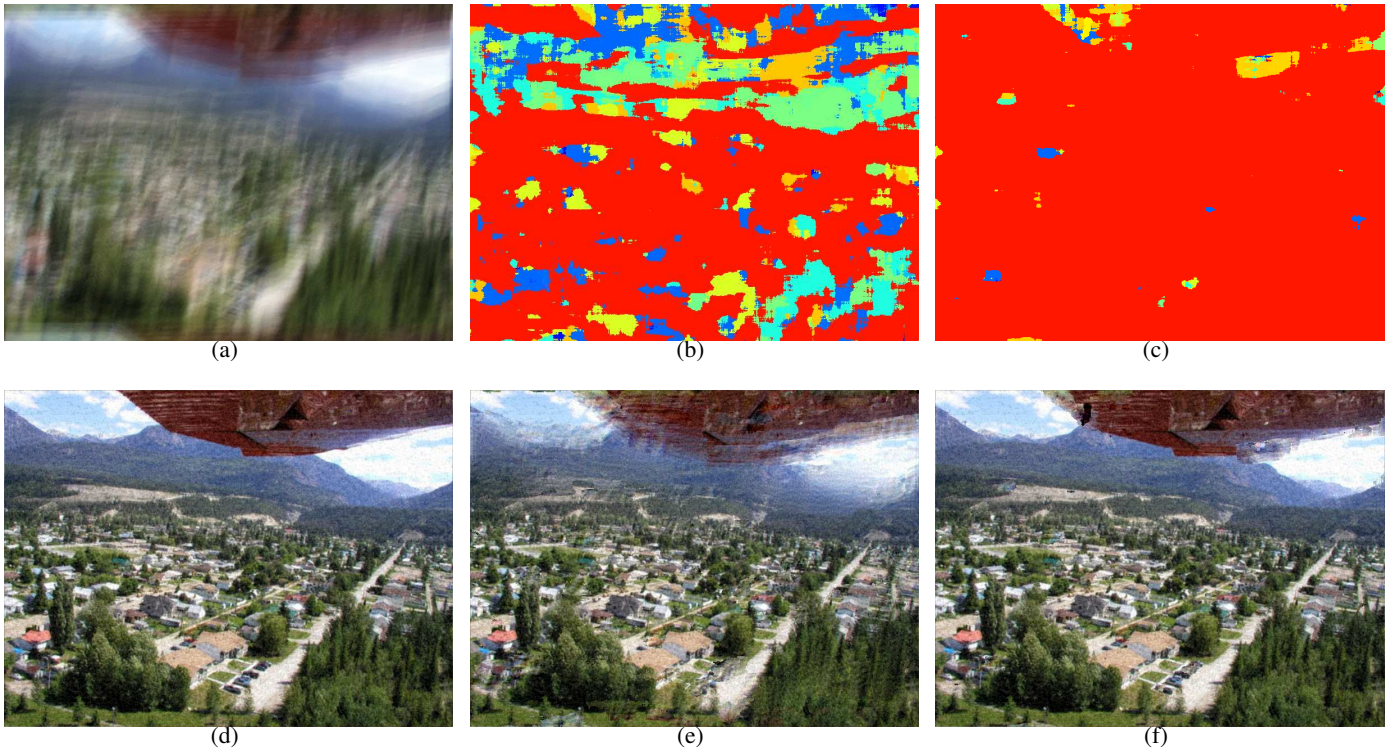


Fig. 44. (a) Input blurry image taken with **a random parabolic camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

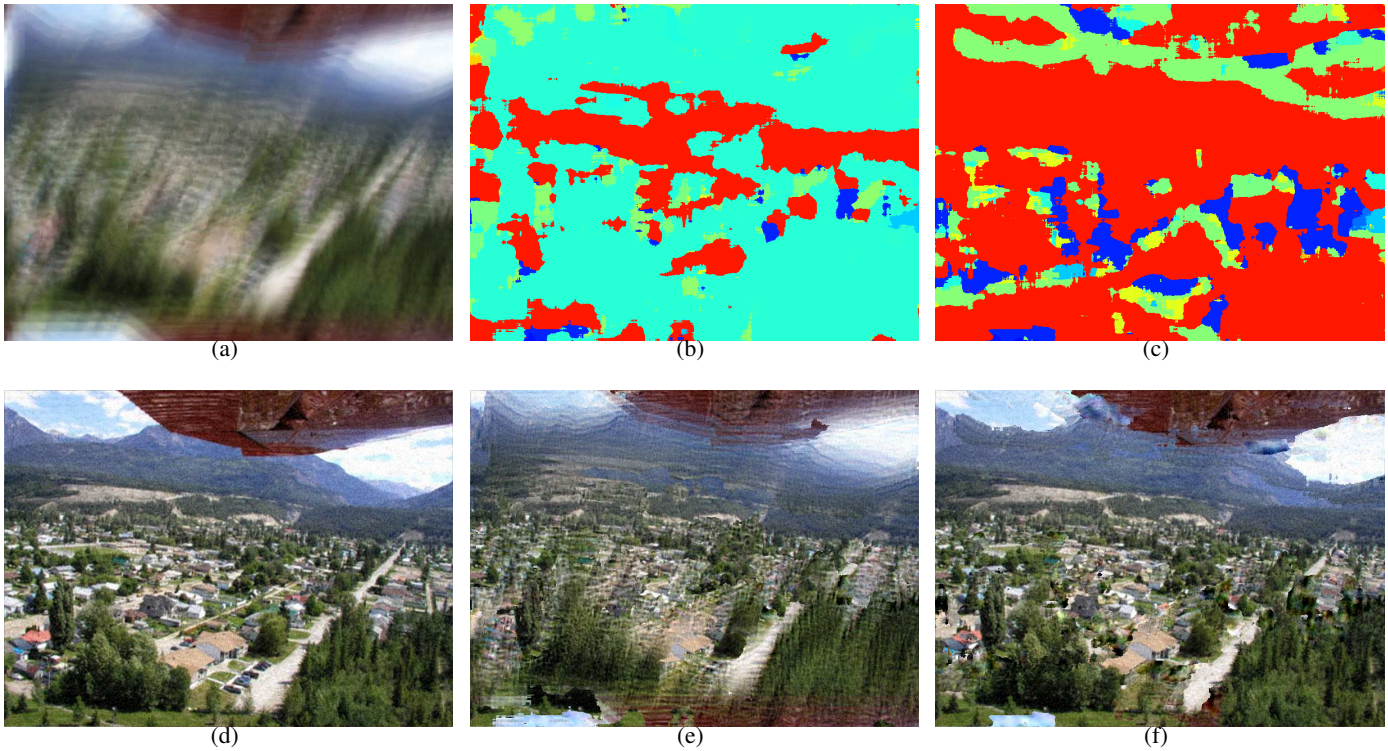
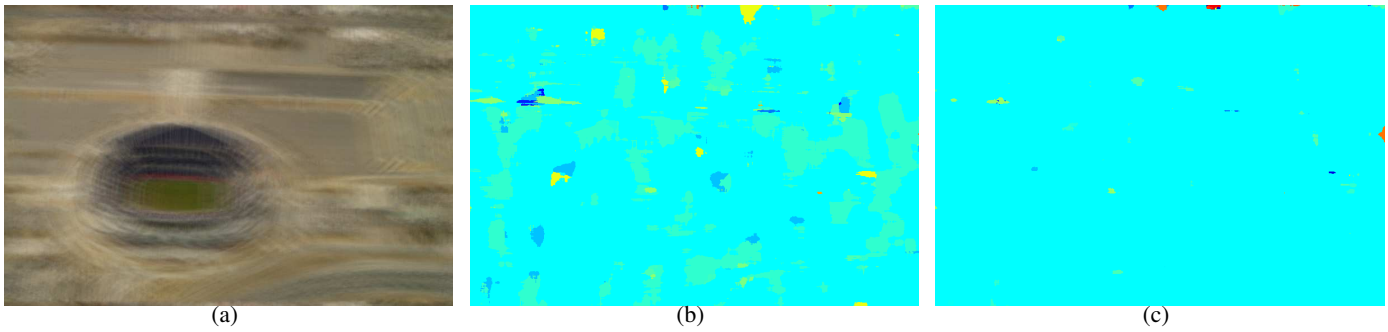
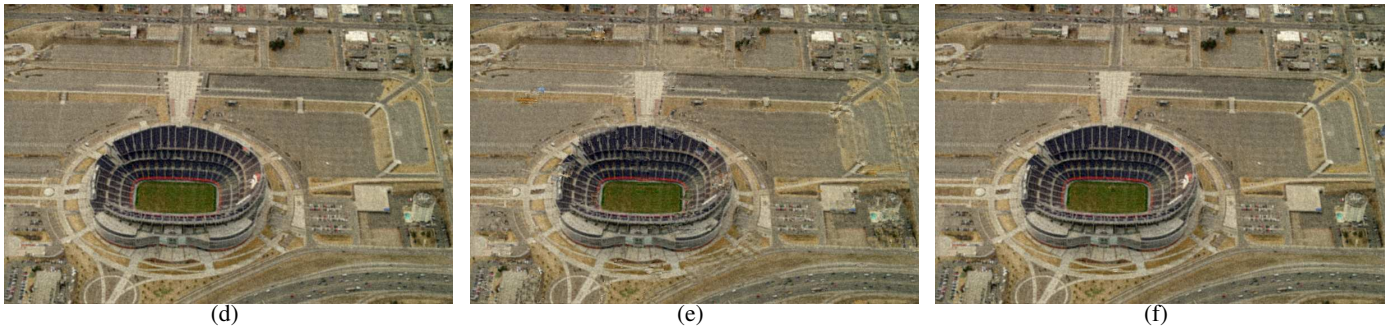


Fig. 45. (a) Input blurry image taken with **a spiral camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

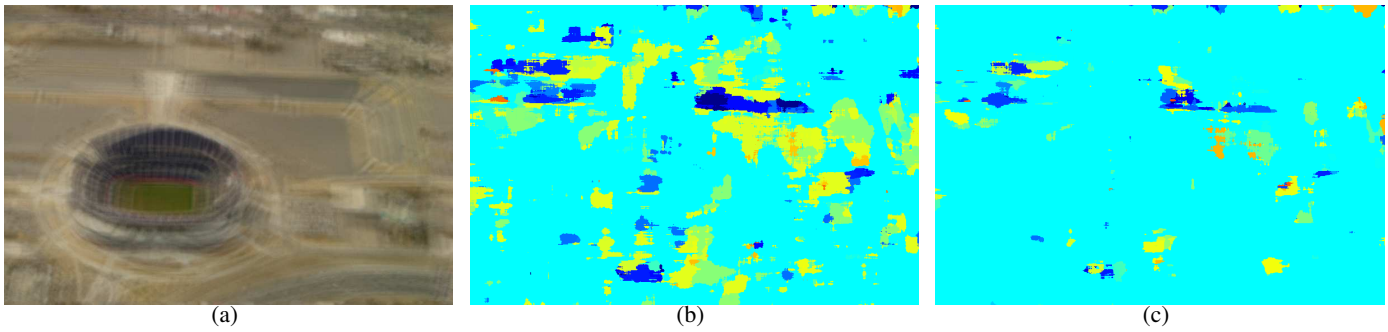


(a) (b) (c)



(d) (e) (f)

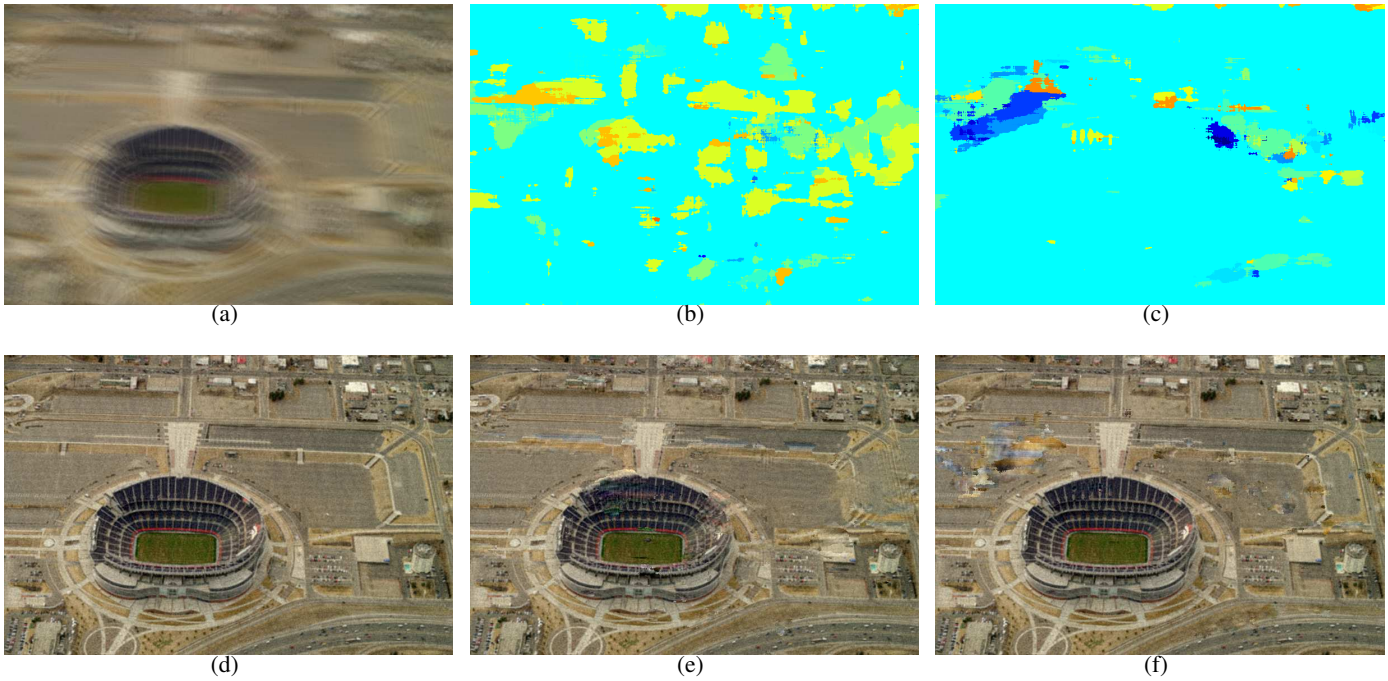
Fig. 46. (a) Input blurry image taken with **a random dot camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

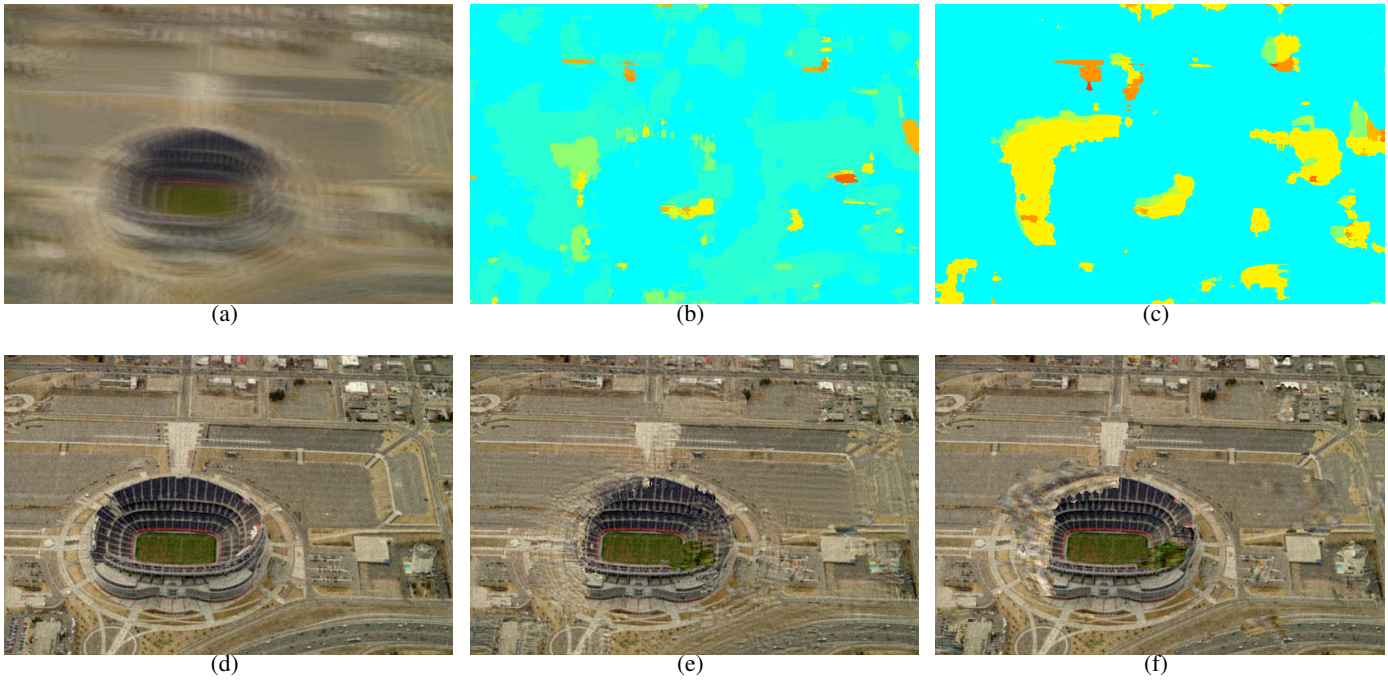
Fig. 47. (a) Input blurry image taken with **a random trace camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

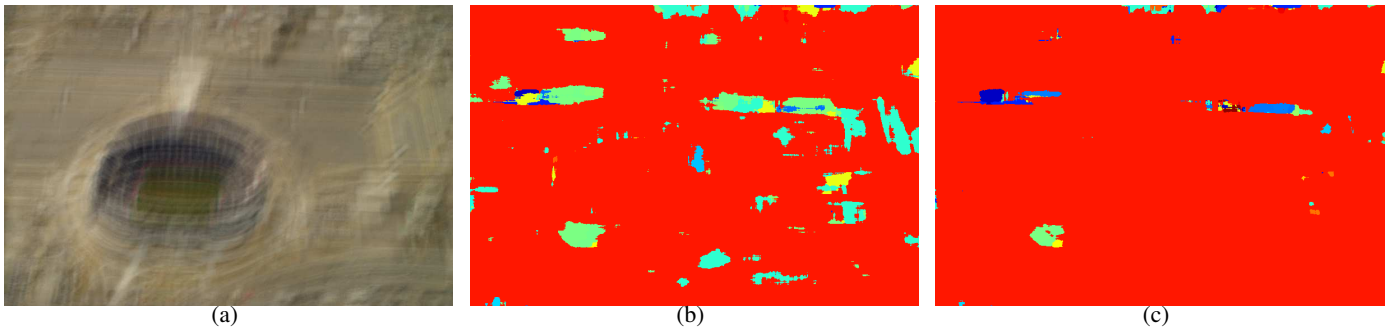
Fig. 48. (a) Input blurry image taken with **a random parabolic camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

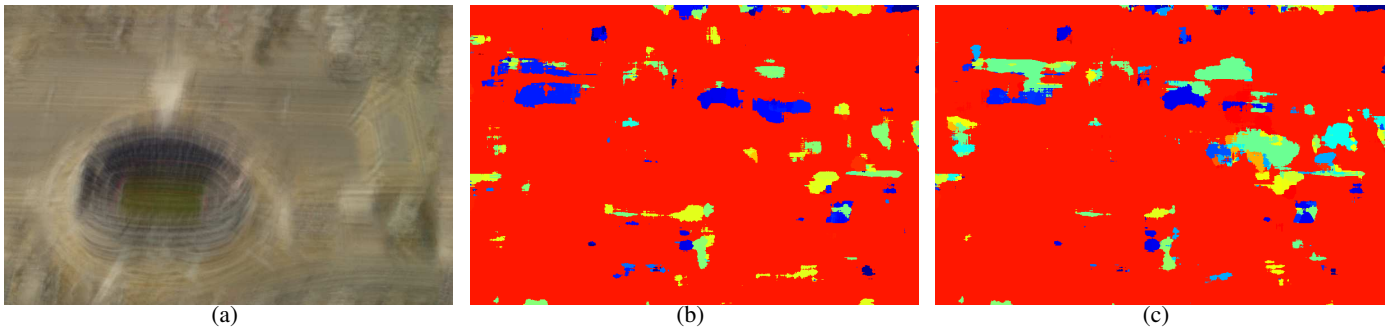
Fig. 49. (a) Input blurry image taken with **a spiral camera** when the object motion is 45. (b) Estimated kernel using the estimation error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

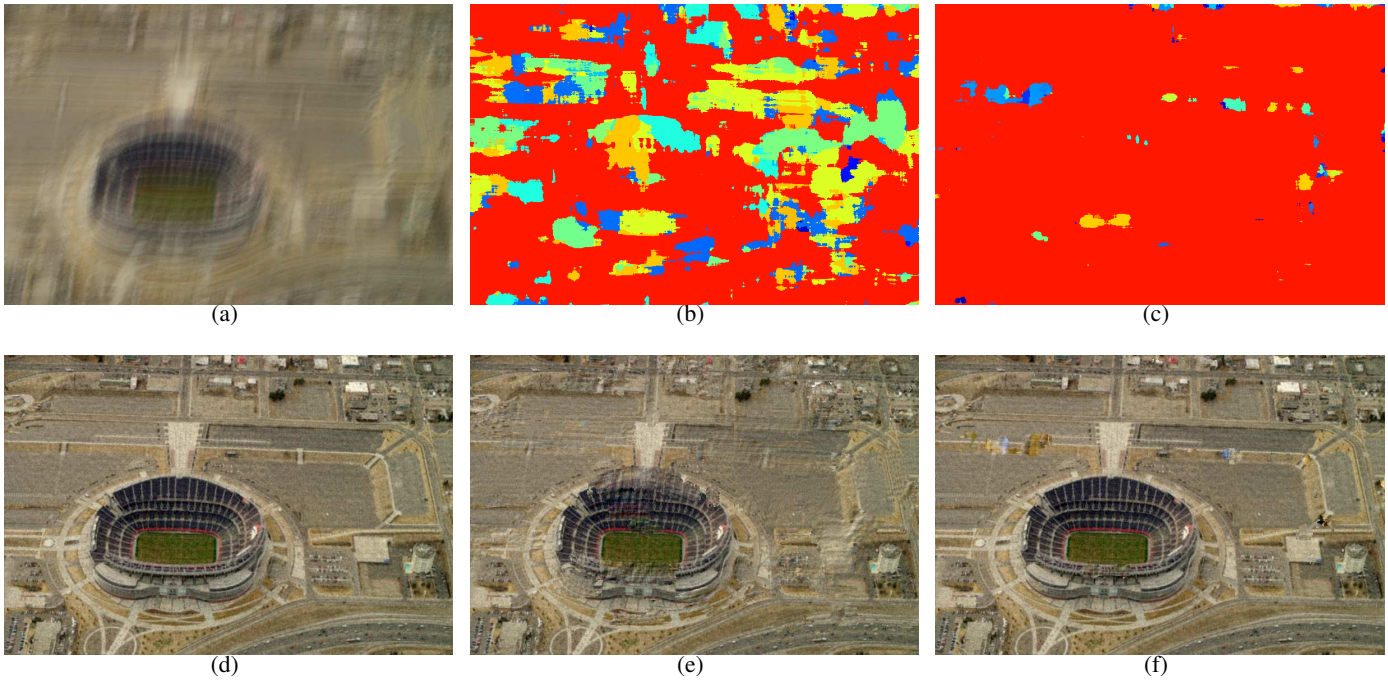
Fig. 50. (a) Input blurry image taken with **a random dot camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

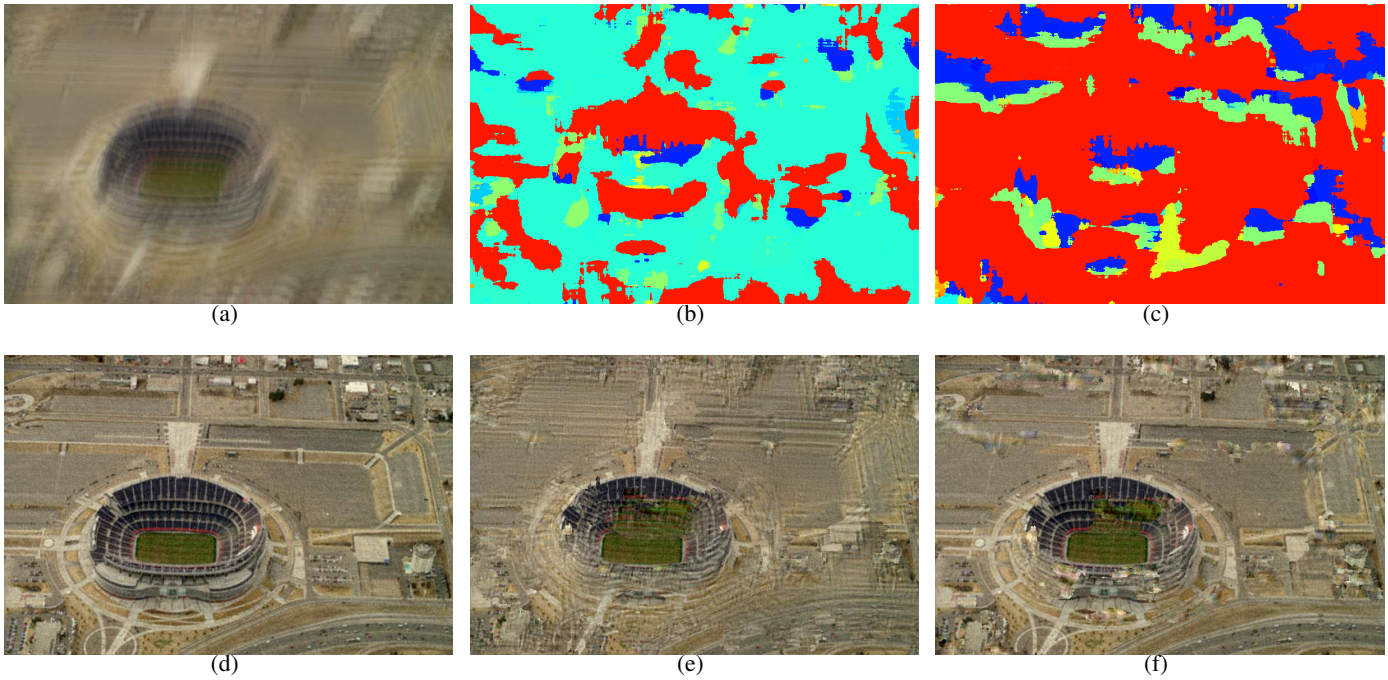
Fig. 51. (a) Input blurry image taken with **a random trace camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

Fig. 52. (a) Input blurry image taken with **a random parabolic camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

(d) (e) (f)

Fig. 53. (a) Input blurry image taken with **a spiral camera** when the object motion is 103. (b) Estimated kernel using the estimation error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (d) Deblurred image with the correct kernel. (e) Deblurred image using the estimation error based measure generated by taking pixels from the corresponding deblurred image. (f) Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

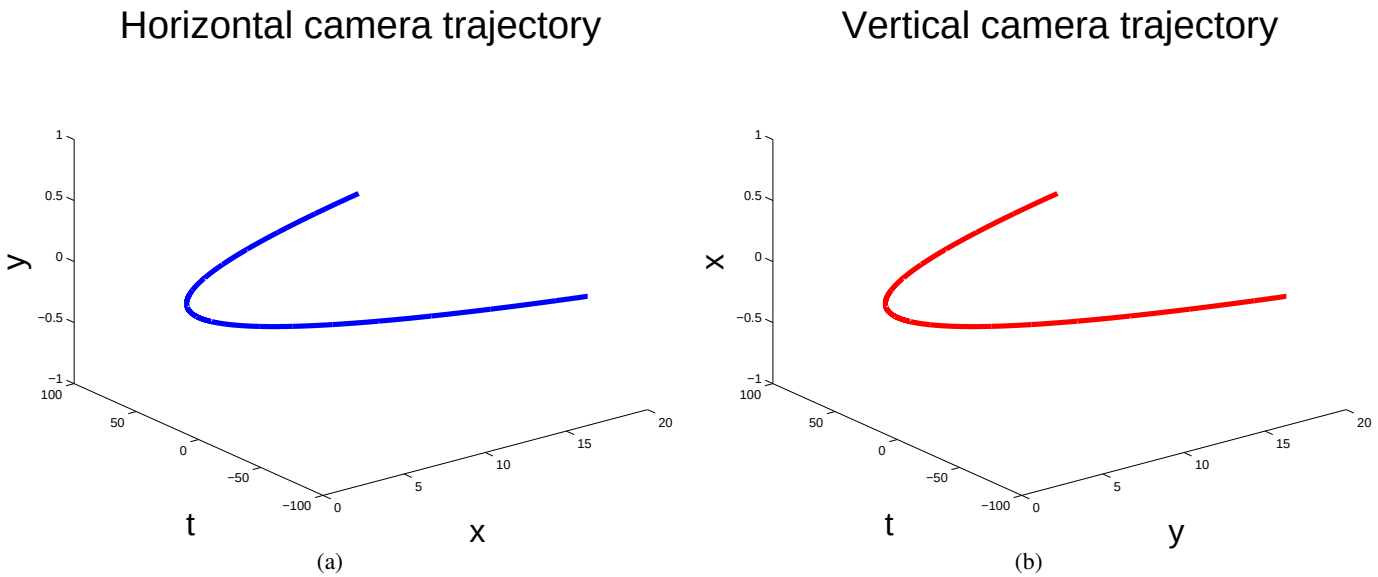


Fig. 54. (a) The trajectory of **a horizontal camera** (b) The trajectory of **a vertical camera**

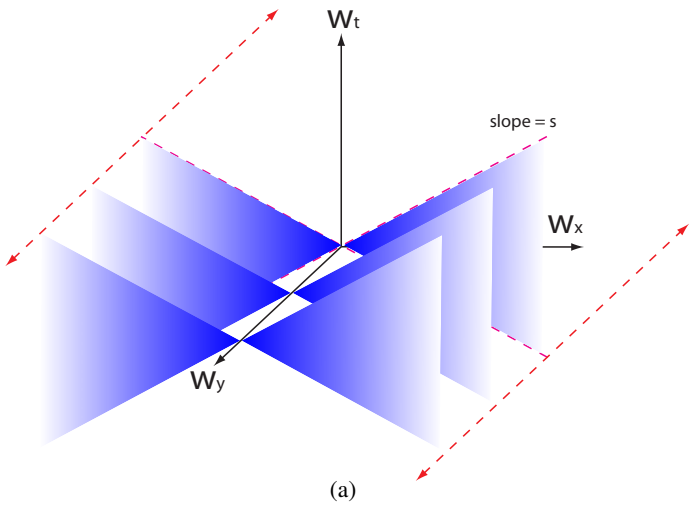


Fig. 55. The 3D Fourier transform of a horizontal parabolic camera. The dotted arrows are used to show that the spectrum is flat along w_x axis.

APPENDIX: TWO ORTHOGONALLY MOVING PARABOLIC CAMERAS

In this appendix, we analyze the performance of two orthogonally moving parabolic cameras, and compare it with the random cameras. Since there are two cameras, we have halved the number of pixels in each camera to appropriately normalize the performance.

F. Camera trajectories and the Fourier Transforms

In this note, we assume that the two parabolic cameras move in 3D space-time volume as shown in Figure 54. The horizontal camera moves along the x axis at $y = 0$, and the vertical camera moves along the y axis at $x = 0$. Let's consider the 3D Fourier transform of a horizontal parabolic camera's trajectory. Since $y = 0$, the Fourier transform is flat along w_y, and it's parabolic in $x - t$ axis so we get a wedged structure in the $w_x - w_t$ axis. This structure is shown in Figure 55. As was studied earlier, as the s_x, the object velocity along the x axis, increases, the slice of the resulting PSF rotates about w_y axis through the origin. Such a slice will have a wedge along the w_y axis.

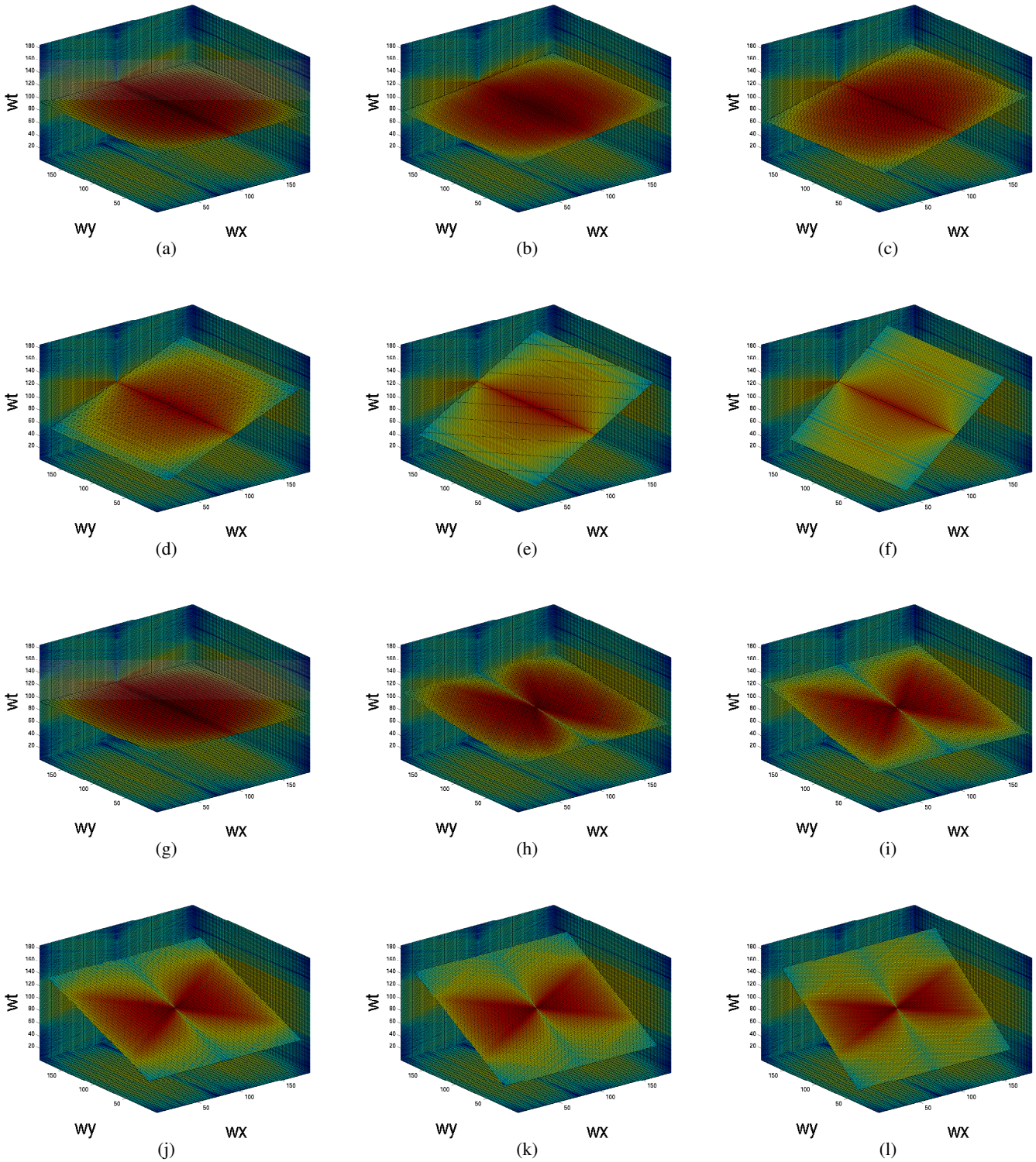
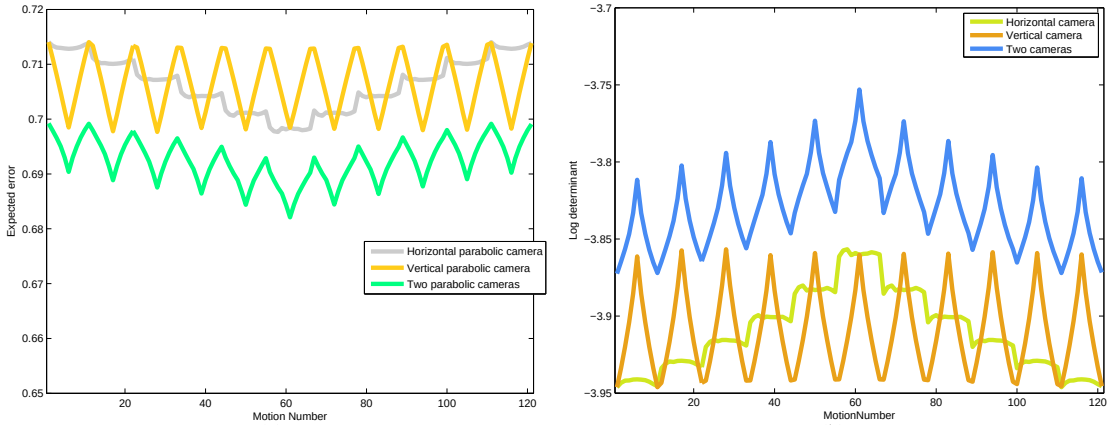


Fig. 56. (a) - (f) A slice of a log-FFT for **the horizontal camera** as we increase the horizontal velocity of the object. Note that the shape of the Fourier spectrum does not change much (recall that this is the 1D case studied in Motion invariant photography.) (g) - (l) A slice of a log-FFT for the horizontal camera as we increase the vertical velocity of the object. As the vertical velocity of the object increases, more constent along the w_y is lost, as expected.

To experimentally prove the theory, we took the Fourier transform of the 3D space-time trajectory of a horizontal camera, and view the slices in Figure 56. Figure 56(a) - (f) show the slices of a log-FFT for the horizontal camera as we increase the horizontal velocity of the object. Note that the shape of the Fourier spectrum does not change



(a) (b)

Fig. 57. (a) The weighted reconstruction error of **a horizontal parabolic camera, a vertical parabolic camera, and two parabolic cameras**. Because each camera loses resolution due to less number of sensor elements, high frequency contents are severely lost, and the weighted reconstruction error - which favors high frequencies - penalyzes that by a large amount. (b) The log determinant of a horizontal parabolic camera, a vertical parabolic camera, and two parabolic cameras.

much (recall that this is the 1D case studied in Motion invariant photography.) Figure 56 (g) - (l) show slices of a log-FFT for the horizontal camera as we increase the vertical velocity of the object. As the vertical velocity of the object increases, more constent along the w_y is lost, as expected.

From these slices, we can see the structure of the 3D Fourier transform (FT) of the trajectory: on the $w_x - w_t$ plane, the FT looks like the wedge structure - with $\frac{1}{w_x}$ fall-off in the w_x direction - studied in Motion Invariant Photography paper; on the $w_x - w_y$ plane, the spectrum is virtually flat along the w_y direction (this is because the y position is an impulse in the spatial domain $\rightarrow$ the frequency content along w_y is flat) modulus the Gaussian, and the spectrum falls off with $\frac{1}{w_x}$ in the w_x direction.

From the symmetry, we can also guess the structure of the 3D FT for the vertical motion camera. The 3D FT for the vertical motion camera trajectory is the same as that of horizontal motion camera with the w_x, w_y axis swap.

When we use two orthogonally moving cameras to capture a 2D motion, we are approximating the inverted double cone (Figure 3) with two wedged structures extending to infinity in each axis. When these wedged structures meet near the origin, the resulting frequencies being covered becomes the two inverted pyramids: we are approximating the inverted double cone with the inverted pyramid, and this is certainly not the optimum in distributing the signal when we want to cover velocities only within S_{mo}. However, one potential benefit in using two cameras is that there isn't any zeros in the spectrum induced by phase cancellations. Another benefit of using two cameras is that we don't need to rely on zeros to identify the correct blur kernel; instead we have the phase constraint between two parabolic cameras, which is stronger than the constraints imposed by zeros.

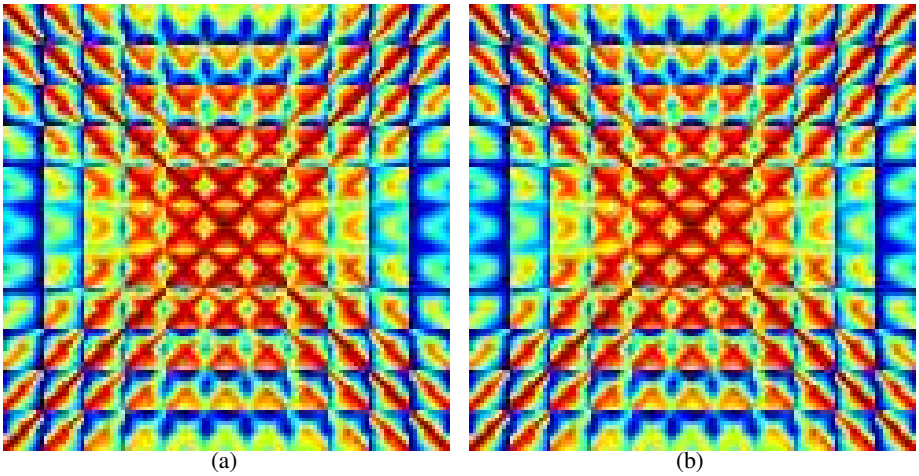
G. The weighted reconstruction error and the log determinants

Figure 57 shows the weighted reconstruction error and the log determinants of a horizontal parabolic camera, a vertical parabolic camera, and two parabolic cameras. Recall that each camera a resolution smaller than that of a single parabolic camera. Reduced resolution implies that much of the high frequency contents are lost, and the weighted reconstruction error, which favors high frequencies, penalyzes the lost high frequency more than the normal reconstruction error. The two parabolic camera setup doesn't perform as well as the random camera setup, which achieves about 0.69 of average weighted reconstruction error.

The log determinants vary quite a bit as the object motion changes, and this is expected from the variation in the weighted reconstruction error. Yet, as we show later, the variation in the log determinant is much less than the estimation error, so we are still able to perform the PSF estimation using just the estimation error.

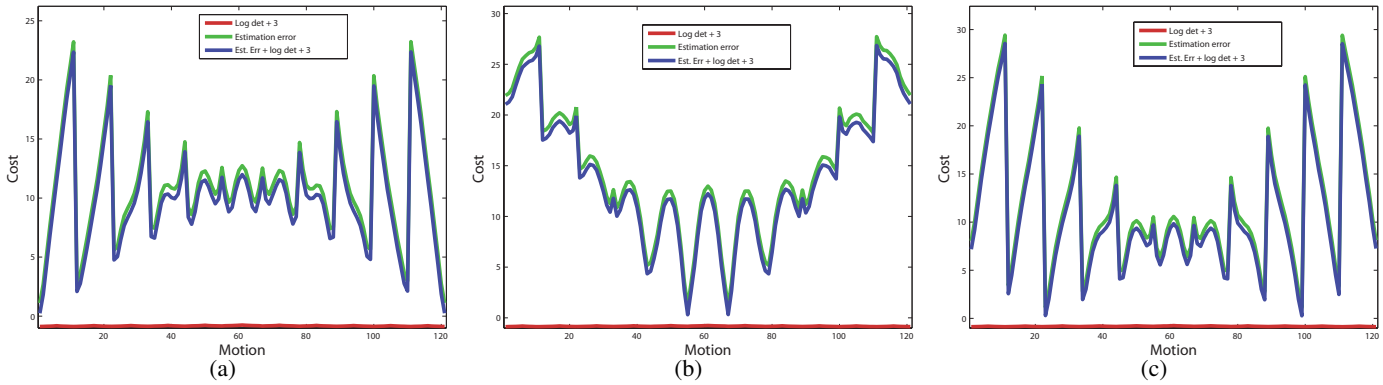
H. The PSF estimation from the input image

As in the random camera analysis, we consider both the whole image-based and the local patch based PSF estimation. 1% noise was added to all blurred images.



(a) (b)

Fig. 58. (a) A confusion matrix using the reconstruction error measure for the **two parabolic cameras**. (b) A confusion matrix using the log likelihood measure for the two parabolic cameras.



(a) (b) (c)

Fig. 59. **Two parabolic cameras**. (a) A slice of confusion matrices at motion 1 (very fast diagonal movement), along with the log det term for each motion kernel. (b) A slice of confusion matrices at motion 55 (very slow horizontal movement), along with the log det term for each motion kernel. (c) A slice of confusion matrices at motion 99 (very fast diagonal movement), along with the log det term for each motion kernel.

1) A whole image based PSF estimation: Figure 58 shows the PSF confusion matrix when we use (a) the estimation error based measure (b) the log likelihood based measure. The confusion matrix is row-wise normalized. The confusion matrix is 121 x 121, and the j^{th} column of the i^{th} row is the score given to the hypothesis that the image originally blurred with i^{th} kernel is identified as being blurred with j^{th} kernel. Higher score means that the algorithm thinks it's more likely.

The PSF estimation is perfect even with just the estimation error based measure. One salient feature in the confusion matrix is that we get two prominent diagonal lines. This can be attributed to the fact that parabolic cameras find it hard to discriminate two motions with same speed, but opposite directions (recall the motion invariance.) Even if we use the log likelihood based measure, such ambiguity is hard to resolve. This attribute is more evident when we look at the slices of the confusion matrix in Figure 59. The estimation error achieves sharp minima at two motions that are of the same speed, but in different directions. When zoomed in, the estimation error has a global minimum at the correct motion. Even if the PSF is misclassified to be the same speed but in different motion, there wouldn't be much visible artifact since the PSF is wrong only by the amount of tail clipping!

Another feature to note is that the variation in the estimation error is much larger than the variation of the log determinant term. In other words, the log determinant term will play minimal role in estimating the correct object motion, so the PSF estimation can be performed solely with the estimation error.

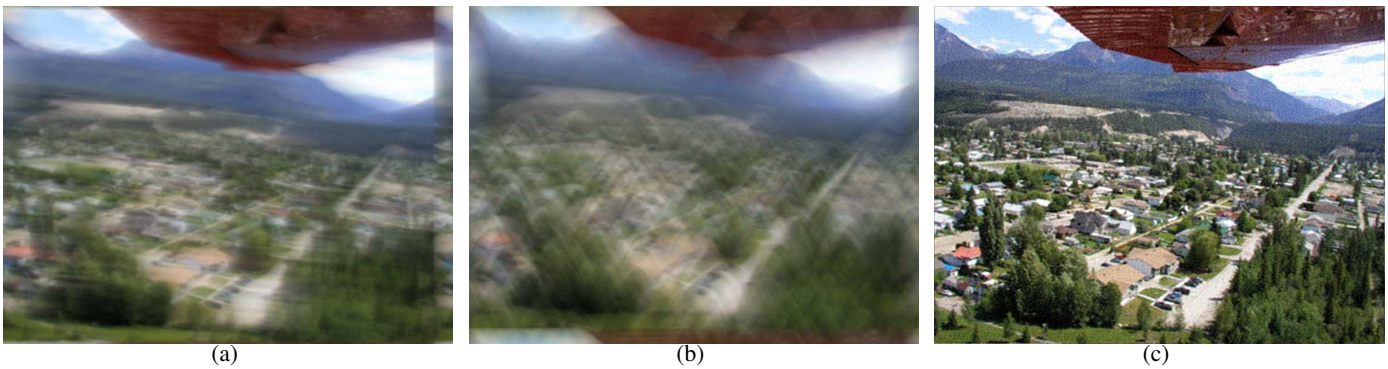
2) PSF estimation from local patches: In this section, we estimate the PSF from the image using local patches. Our test set is two images, each blurred with the same PSF across the whole image, and our goal is to correctly

identify the PSF that blurred the input images from a pool of 121 blur kernels. The patch size is 41 x 41 in this experiment.

The PSF estimation results are shown in the subsequent pages. One thing to notice is that the two parabolic cameras outperform other random cameras in estimating the correct PSF from local patches, even the random dot camera with disconnected paths! This can be attributed to the phase constraint in the Fourier coefficients, as we mentioned earlier.

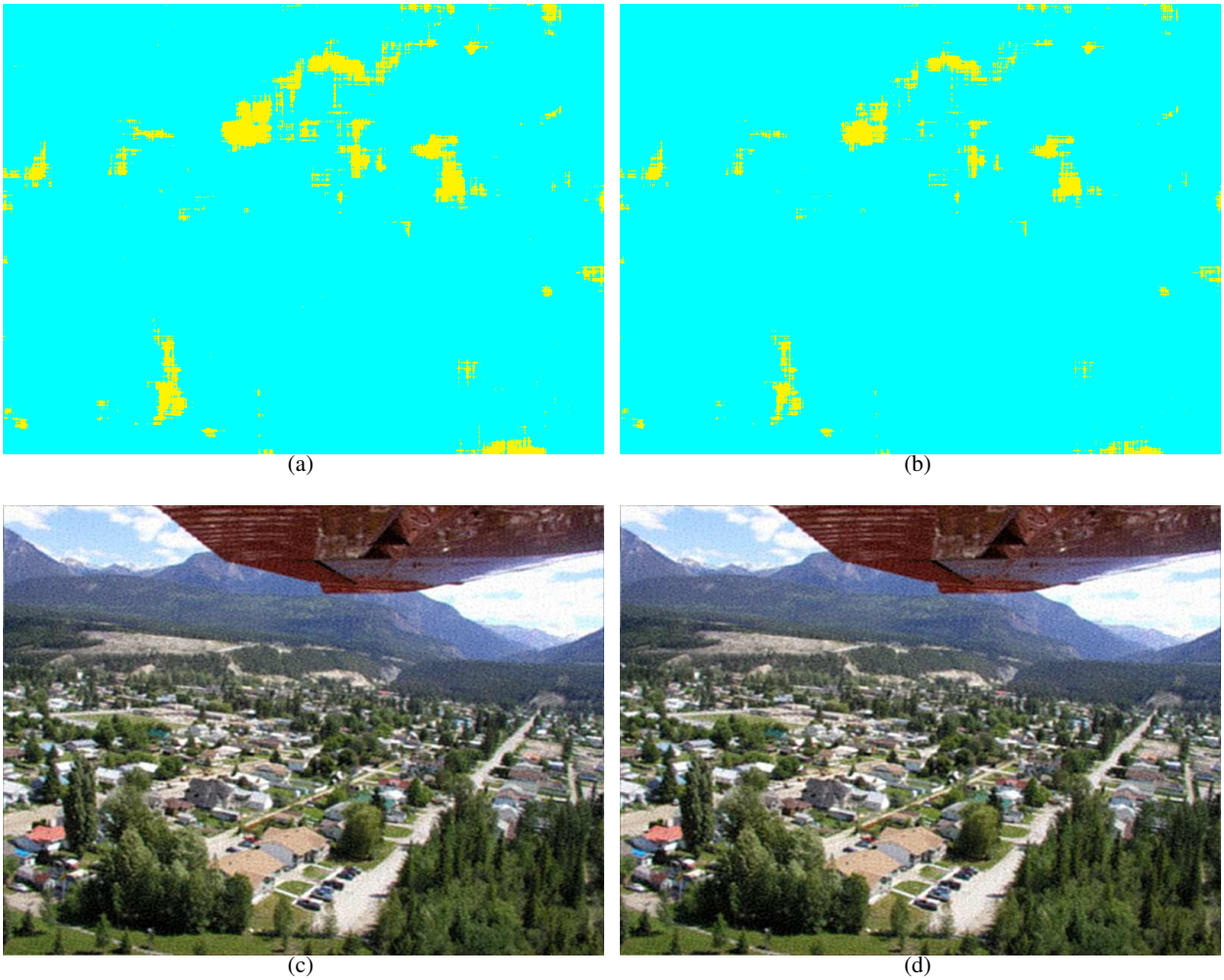
The misclassified islands in the layers is much smaller than other random cameras, so we may be able to eliminate most of the misclassified islands using the graph-cut algorithm. The log determinant term does help in correcting some of the islands, and thus we should use the log determinant term when estimating the PSF in practice.

Another interesting, yet expected, result is that even if the PSF estimation is incorrect, the algorithm always opts for the same speed in the opposite direction. This is expected from the confusion matrix and its slices. Therefore, even if the PSF is misclassified, there isn't any visible artifact at the output since the PSF is wrong only by the amount of tail clipping!



(a) (b) (c)

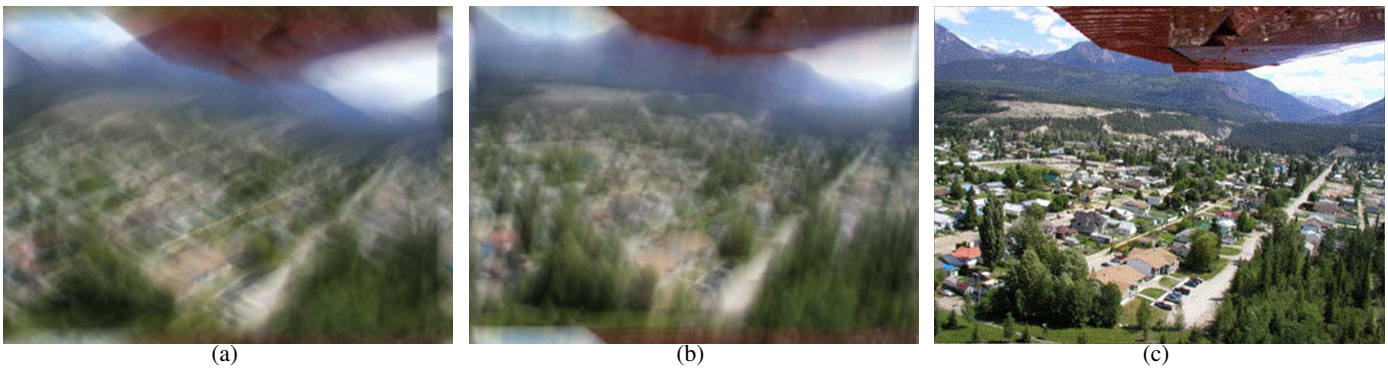
Fig. 60. **Two parabolic cameras**. (a) Input blurry image taken with a random dot camera when the object motion is 45. (b) Deblurred image with the correct kernel.



(a) (b)

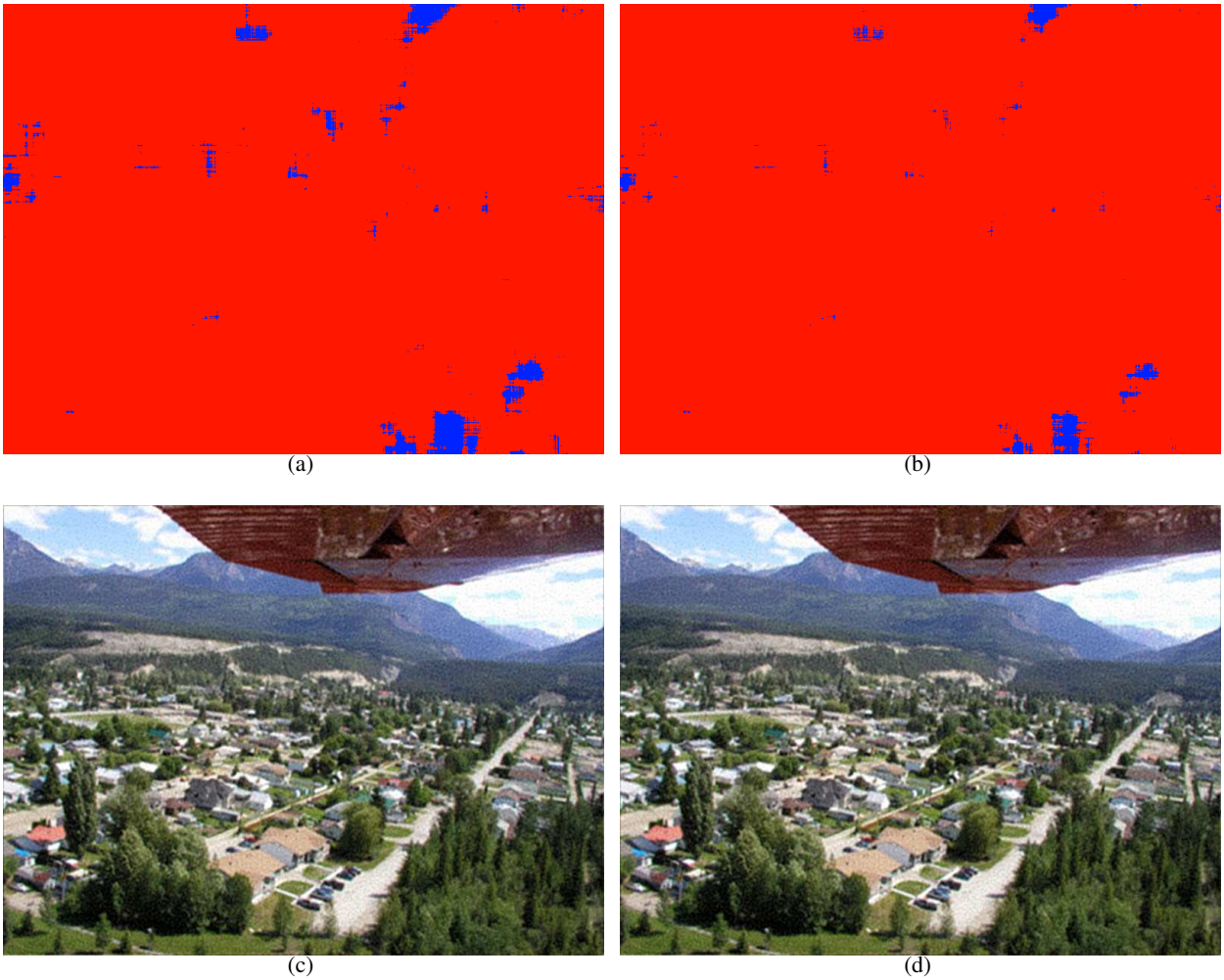
(c) (d)

Fig. 61. **Two parabolic cameras**. (a) Estimated kernel using the reconstruction error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (b) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (c) Deblurred image using the reconstruction error based measure generated by taking pixels from the corresponding deblurred image. (d)Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.



(a) (b) (c)

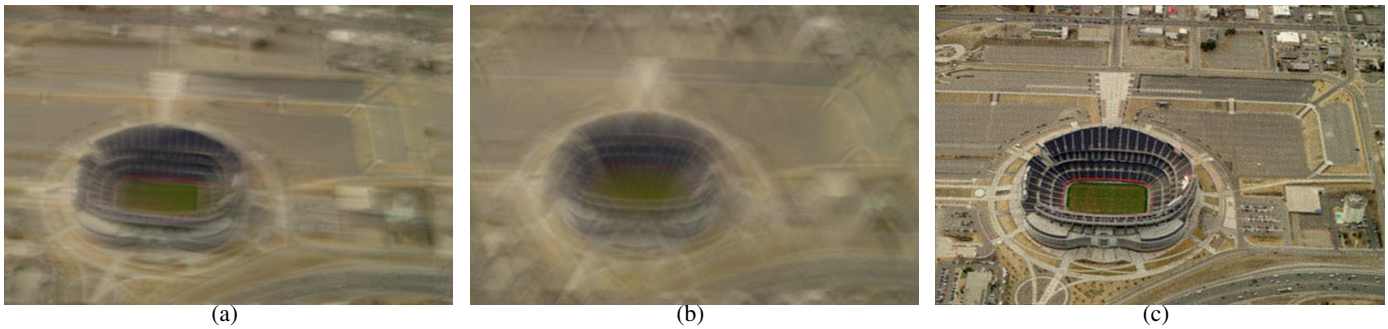
Fig. 62. **Two parabolic cameras**. (a) Input blurry image taken with a random dot camera when the object motion is 103. (b) Deblurred image with the correct kernel.



(a) (b)

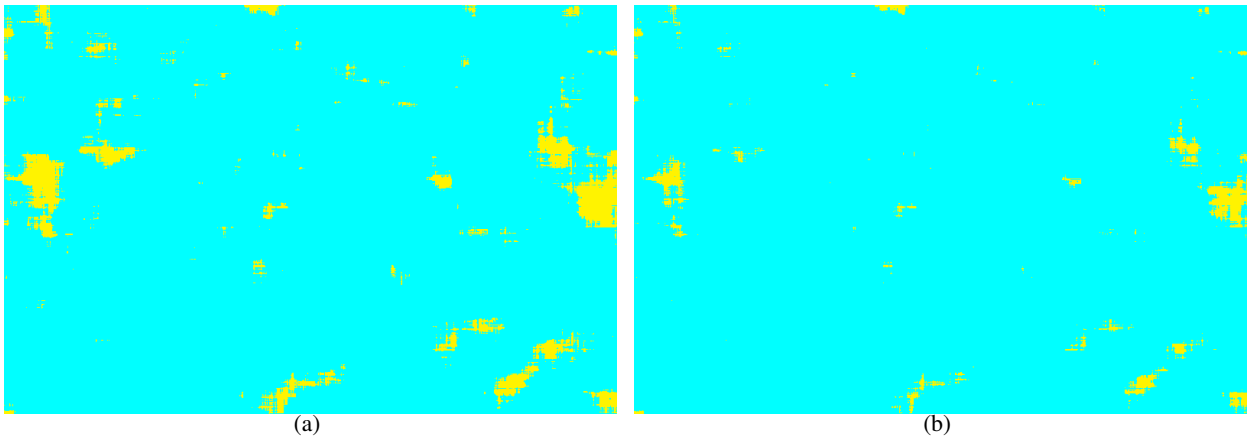
(c) (d)

Fig. 63. **Two parabolic cameras**. (a) Estimated kernel using the reconstruction error based measure. The correct kernel is light red, and the zero motion explanation is light green. (b) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. (c) Deblurred image using the reconstruction error based measure generated by taking pixels from the corresponding deblurred image. (d)Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

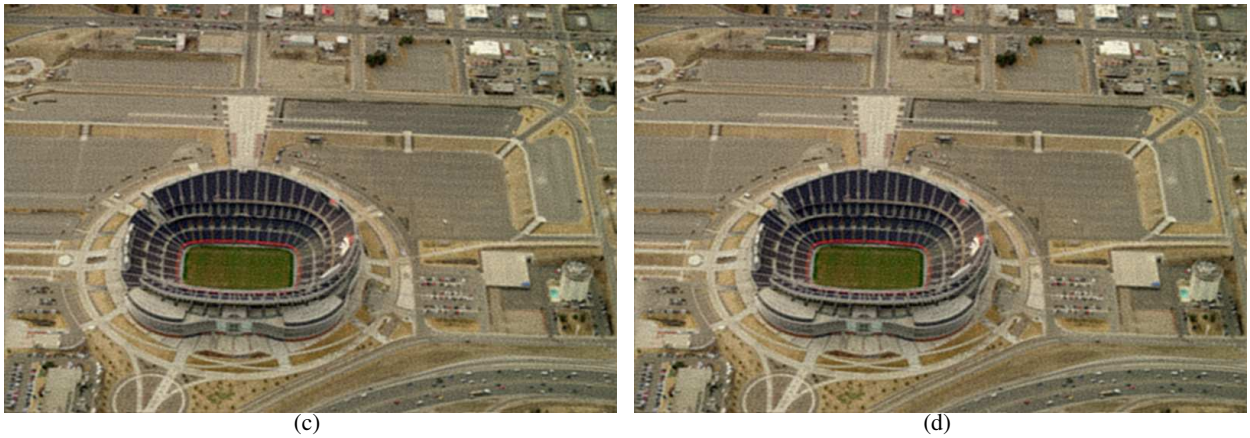


(a) (b) (c)

Fig. 64. **Two parabolic cameras**. (a) Input blurry image taken with a random dot camera when the object motion is 45. (b) Deblurred image with the correct kernel.

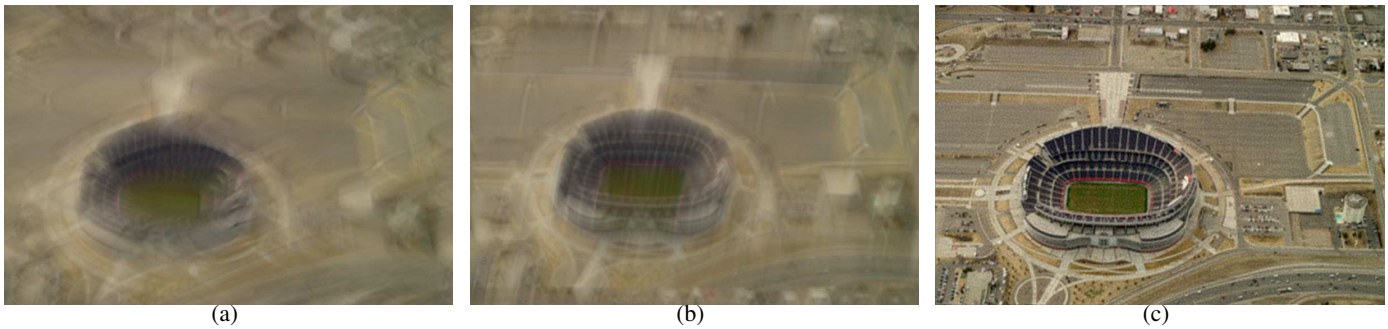


(a) (b)



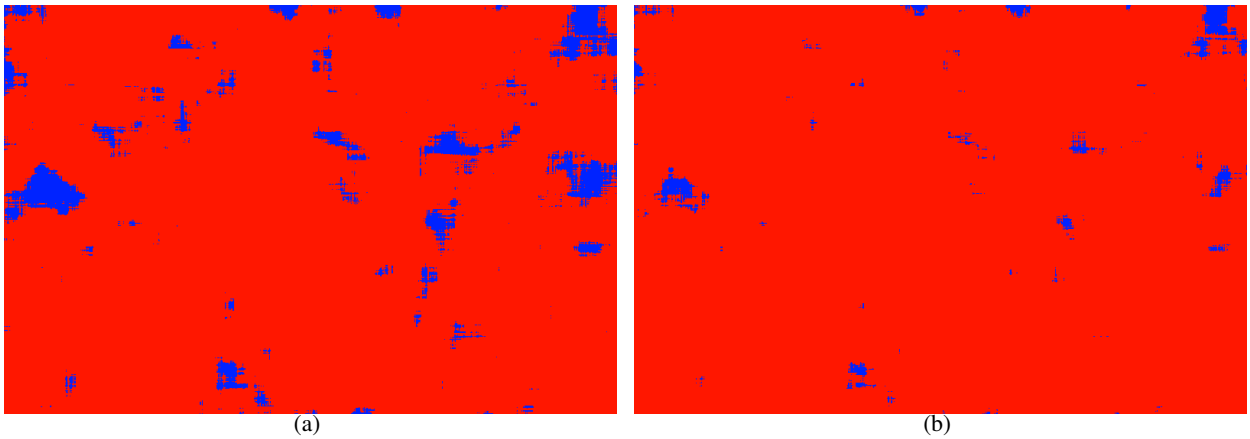
(c) (d)

Fig. 65. **Two parabolic cameras**. (a) Estimated kernel using the reconstruction error based measure. The correct kernel is light blue, and the zero motion explanation is light green. (b) Estimated kernel using the log likelihood error based measure. The correct kernel is light blue, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is yellow. (c) Deblurred image using the reconstruction error based measure generated by taking pixels from the corresponding deblurred image. (d)Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.

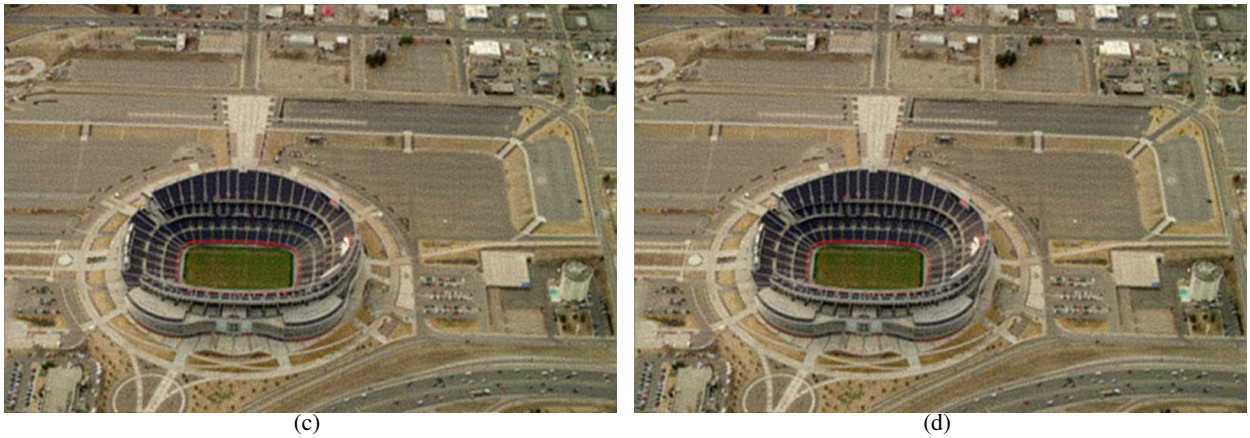


(a) (b) (c)

Fig. 66. **Two parabolic cameras**. (a) Input blurry image taken with a random dot camera when the object motion is 103. (b) Deblurred image with the correct kernel.



(a) (b)



(c) (d)

Fig. 67. **Two parabolic cameras**. (a) Estimated kernel using the reconstruction error based measure. The correct kernel is light red, and the zero motion explanation is light green. (b) Estimated kernel using the log likelihood error based measure. The correct kernel is light red, and the zero motion explanation is light green. The kernel that corresponds to the same speed, but in the opposite direction is blue. (c) Deblurred image using the reconstruction error based measure generated by taking pixels from the corresponding deblurred image. (d)Deblurred image using the log likelihood based measure generated by taking pixels from the corresponding deblurred image.