An Introduction to Super-Resolution Imaging

Jonathan Simpkins and Robert L. Stevenson

University of Notre Dame, 275 Fitzpatrick Hall, Notre Dame, IN 46556, USA

0. THE BIG PICTURE

This chapter is presented as an introduction to super-resolution imaging, intended for beginning graduate students, advanced undergraduate students, and researchers who are interested in learning about super-resolution.

"Super-resolution" is the term used to denote the subset of imaging processing that tries to estimate a high-resolution image of a scene, given a set of low-resolution observations (Fig. 1). Colloquially, when super-resolution researchers try to describe what they do to family, friends, and loved ones, we can use the example from the TV show CSI: whenever someone on CSI looks at low-resolution security camera footage, pushes the magic "Enhance" button, and then suddenly the footage is crystal clear, that character is using super-resolution. The programs that we write in super-resolution research are like the real-life version of the "Enhance" button.

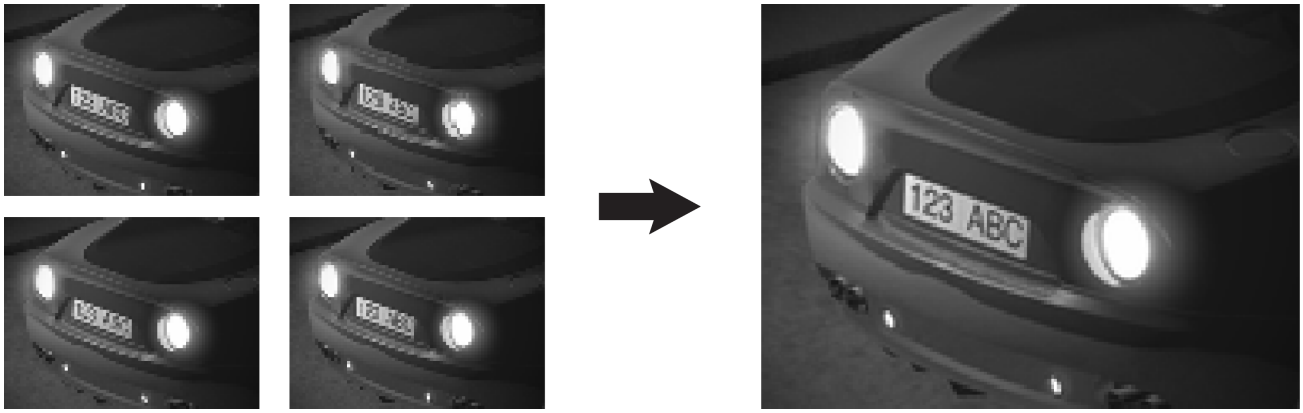

Figure 1. A set of low-resolution observed images (left) and the computed super-resolution estimate (right).

Observed images can be of insufficient resolution or clarity for several reasons.[1] The camera might be required to be far away from the subject (e.g. satellite or high-altitude imaging), the images may have been recorded without prior knowledge of which part of the image will be important (e.g. security footage of a crowd or a large area), or the application might dictate that high-resolution imaging sensors are simply too expensive or not feasible for a given size (e.g. cell phone cameras, although this is becoming less true as technology advances).

Over the course of this chapter, we are going to consider various aspects of the super-resolution problem, and build up from simple examples into more and more realistic approaches. There are 8 different Mathematica code examples (tested in Mathematica v. 8.0.4.0), demonstrating examples of everything from simulating low-resolution images, to computing super-resolution estimates for monochrome and color images under different assumptions, to computing motion estimates from a set of low-resolution observations. The code throughout is formatted like C code, to increase readability for readers less familiar with Mathematica coding conventions. We understand that this approach is inefficient (both in terms of performance and in terms of code length), but clarity of approach is our goal here, and we hope that our examples are helpful even to readers who are developing in a different environment (e.g. OpenCV or Matlab).

Further author information: (Send correspondence to J.D.S)
J.D.S: E-mail: jsimpkin@nd.edu
R.L.S.: E-mail: rls@nd.edu

1. FORMATION OF LOW-RESOLUTION IMAGES

In order to tackle the problem of estimating a super-resolved image from a set of low-resolution observations, we begin by looking at how the low-resolution images are formed.

Typically, the set of low-resolution images are recorded sequentially by a single camera (e.g. frames from a video sequence).[2] There is usually motion between any pair of images in the observation set (either due to camera motion or motion of the subject), and the image of the scene is blurred in some way as it passes through the camera system. The final recorded image is sampled at a relatively low spatial frequency, and additive noise corrupts the image observation.

This observation model can be summarized for each observation with a set of three linear operators and the addition of a noise term. Suppose that there is a high-resolution (HR) image $\mathbf{Z}$ of a scene, and that this image is what our super-resolution (SR) algorithm is trying to estimate.[3] It is typical to model each low-resolution (LR) observation $\mathbf{y}_i$ of the HR image as:

$$\mathbf{y}_i = \mathbf{D}_i\mathbf{F}_i\mathbf{H}_i\mathbf{Z} + \mathbf{n}_i \tag{1}$$

where $\mathbf{D}_i$ represents downsampling of the i'th image, $\mathbf{F}_i$ represents the geometric transformation of the i'th image relative to the 1st image of the set, $\mathbf{H}_i$ represents the effect of blur on the i'th image, and $\mathbf{n}_i$ is the additive noise term.[4]

In this chapter, we will impose a few restrictions on these operations, just as an introduction to the subject. We assume that the effect of blur is spatially-invariant across the image plane, that the blur kernel is known to the SR estimation algorithm, and that it is consistent across all LR observations in the set (we will discuss blurred images in more detail in Section 5). We assume (as does nearly every paper on super-resolution) that the noise is white Gaussian noise, with the same noise variance across all LR observations. Finally, we assume that the geometric transformation $\mathbf{F}_i$ of each image, also sometimes referred to as the "motion estimate" of the image, is constrained to global translation (we will discuss motion estimation in Section 7).

Under these assumptions, there are essentially five parameters for generating a set of LR observations: the magnification from the LR images to the HR ground-truth image, the number of observation images, the translations of those observations (along each axis), the blur kernel, and the standard deviation of the observation noise. In the Mathematica examples throughout the chapter, we will define these parameters as follows:

Program 1 Definition of the parameters for a set of LR observations.

```
(* Define the SR magnification *)
S = 3;

(* Define the number of LR images, and their offsets *)
NImages = 4;
dx = {0, 0, 2, 1};
dy = {0, 1, 0, 2};

(* Define the std deviation of the noise *)
NoiseStd = 5/255;

(* Define the blur kernel *)
K = {{1, 2, 1}, {2, 4, 2}, {1, 2, 1}}/16;
```

A few quick comments on these definitions: in the code in the rest of this chapter, we impose that the magnification S is an integer (e.g. a magnification of 2.74 might be realistic in a real-world application, but we choose

to work with a magnification of 3 instead). Similarly, we will constrain the translational offsets of the image to be integers (integers on the scale of the HR image, which will translate to fractional offsets on the scale of the LR images). Finally, the standard deviation $NoiseStd$ of the observation noise is defined on a scale where black is 0.0, and white is 1.0. So in the definition in Program 1, the noise has a standard deviation of 5 in the typical 8-bit color depth (where values range from 0 to 255).

The magnification S is used when computing the downsampling grid from the HR image to the LR image: the LR image is composed of every S'th sample along each axis (Fig. 2). We notice from Eqn. 1 that because the downsampling operator $\mathbf{D}_i$ is the last operator to apply to the HR image, downsampling occurs on the blurred, shifted version of the desired HR image.

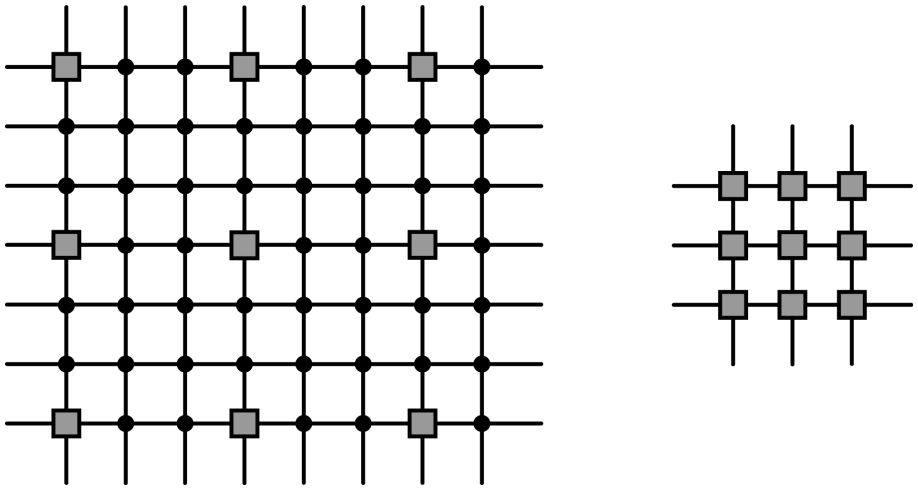

Figure 2. The downsampling grid for a monochrome image.

Given the parameter definitions, it's relatively straight-forward to simulate the set of LR observations. The code in Program 2 will be referenced throughout this chapter, because with the exception of color images (discussed in Section 6), this LR observation model will be the same regardless of our reconstruction approach.

Program 2 Importing an HR ground-truth image, and simulating the LR observations

```
(* Import the image, and force it to be grayscale *)
InputImage = ColorConvert[Import["TestImage.jpg"], "Grayscale"];

(* Make sure the width and height are divisible by S *)
ImDim = ImageDimensions[InputImage];
W = ImDim[[1]] - Mod[ImDim[[1]], S];
H = ImDim[[2]] - Mod[ImDim[[2]], S];
InputImage = ImageCrop[InputImage, {W, H}];

(* Blur the HR image *)
BlurIm = ImageConvolve[InputImage, K];

(* Simulate the set of LR observations *)
SetOfImages = Array[0, {NImages}];
For[ImCount = 1, ImCount <= NImages, ImCount++,
  SmallImArray = ConstantArray[0, {H/S, W/S}];
  For[i = 0, i < H/S, i++,
    For[j = 0, j < W/S, j++,
      px = j*S + dx[[ImCount]] + 0.5;
      py = i*S + dy[[ImCount]] + 0.5;
      SmallImArray[[i+1, j+1]] =
        ImageValue[BlurIm, {px, py},
          Resampling -> "Bilinear",
          Padding -> "Fixed"]
        + NoiseStd*RandomVariate[NormalDistribution[0,1]];
    ];
  ];
  SetOfImages[[ImCount]] = ImageReflect[Image[ SmallImArray ]];
];
```

We encourage the reader to experiment with this code, and try simulating sets of LR observations under different conditions, and for different HR ground-truth images. In particular, it is interesting to look at what the LR observations look like when the magnification S is large (e.g. 3 or 4), and the blur is negligible (e.g. setting $K = \{\{1\}\}$ is equivalent to simulating LR observations with no blur). An example of these kind of LR observations is shown in Fig. 3.


Figure 3. The HR ground-truth image (left), and a simulated LR observation (right) with a magnification S of 3 and no blur or noise.

Readers that have experience with signal processing might notice that the LR image in Fig. 3 show signs of aliasing (most notably on the building on the left-hand side of the image). This aliasing is due to the fact that the downsampling $\mathbf{D}$ is taking place at a lower spatial frequency than the Nyquist frequency of the blurred HR image. In a normal imaging application, we might be disappointed if a camera gave us this LR observation, because the aliasing effect is so visually unappealing. But as we'll see in the next section, this aliasing is *exactly* what makes it possible to use the set of LR observations to compute an accurate SR estimate.

2. SUPER-RESOLUTION: ORIGINS

Tsai and Huang in 1984* published what is generally considered to be the paper that started super-resolution.[5] Their key observation followed from the fact that if an imaging sensor under-samples the two-dimensional frequency content of an image, the recorded image will display aliasing.[6]

If $F(u, v)$ is the 2D continuous Fourier transform (CFT) of the image plane, $F_{m,n}$ is the 2D discrete FT (DFT), and T_1, T_2 are the sampling intervals along the x- and y-axis, respectively, then we can express the DFT as an infinite summation of samples from the CFT.

$$F_{m,n} = \frac{1}{T_1 T_2} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F\left(\frac{m}{MT_1}, \frac{n}{NT_2}\right) \tag{2}$$

and where M, N are the number of samples taken along each axis of the image (i.e. an image with MN recorded pixels). As T_1 and T_2 become large (as the sampling frequency decreases, as with a lower-megapixel sensor of the same size), the copies of the image spectra in the frequency domain move closer together, and when sampling becomes too sparse, the sprectra overlap (Fig. 4).

*And yes, we acknowledge the irony that the subset of image processing that has allowed greater ability for remote sensing and surveillance began in 1984.

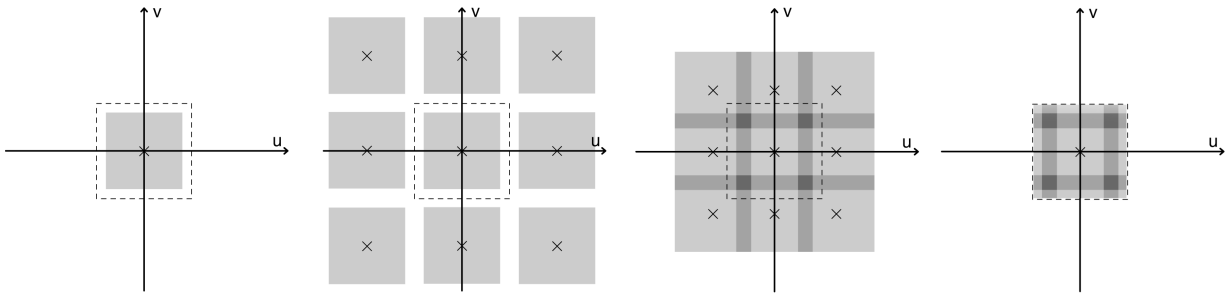

Figure 4. The frequency content of a bandlimited image (far left), the result of sampling the image above the Nyquist frequency (center left), the result of sampling the image below the Nyquist frequency (center right), and the observed aliased frequency content (far right).

When overlap of spectra happens, we might think of the image frequency content as irrevocably ruined, and this would be the point in the signal processing class where it's normally emphasized that you should sample above the Nyquist frequency to avoid aliasing. But, this aliasing turns out to be a way of essentially packing higher-frequency data into a sparsely-sampled image. The trick is getting the aliased information back out.

Let's look at, for example, what happens when we undersample the image by a factor of 2 (i.e. if we had 4x as many pixels, we would notice no aliasing in the image). Let $F_{m,n}$ denote the DFT of the densely sampled HR image (of size $2M$x$2N$), and $G_{m,n}$ denote the actual observed DFT from the sparsely sampled LR image (of size MxN). We can write every aliased frequency sample as the complex sum of 4 frequency samples from the non-aliased DFT:

$$G_{m,n} = F_{m,n} + F_{m+M,n} + F_{m,n+N} + F_{m+M,n+N} \tag{3}$$

Now suppose we have several LR observations, and we know the relative translation of each one (translation on the scale of the densely sampled HR image). If for a shift of (x, y) we denote the LR (aliased) frequency sample as $G^{x,y}_{m,n}$ and the non-aliased HR frequency sample as $F^{x,y}_{m,n}$, then it's true that:

$$G^{x,y}_{m,n} = F^{x,y}_{m,n} + F^{x,y}_{m+M,n} + F^{x,y}_{m,n+N} + F^{x,y}_{m+M,n+N} \tag{4}$$

$$= F^{0,0}_{m,n} \exp\left(-j2\pi\left(\frac{mx}{2M} + \frac{ny}{2N}\right)\right) + F^{0,0}_{m+M,n} \exp\left(-j2\pi\left(\frac{(m+M)x}{2M} + \frac{ny}{2N}\right)\right) \tag{5}$$

$$+ F^{0,0}_{m,n+N} \exp\left(-j2\pi\left(\frac{mx}{2M} + \frac{(n+N)y}{2N}\right)\right) + F^{0,0}_{m+M,n+N} \exp\left(-j2\pi\left(\frac{(m+M)x}{2M} + \frac{(n+N)y}{2N}\right)\right) \tag{6}$$

The important thing to notice here is that each of the 4 aliased frequency samples $F_{m\pm M,n\pm N}$ experiences a different phase shift for the same spatial shift. The observed frequency response $G_{(m,n)}$ is then a complex linear function of 4 unknowns (the aliased frequency samples $F^{0,0}_{m\pm M,n\pm N}$). And in this insight is where super-resolution becomes possible: what if you have 4 LR observations, each with a different shift (x, y)? If the shifts are chosen correctly, it would be possible to form a system of linearly independent equations, which could be solved to get the 4 aliased frequency samples $F^{0,0}_{m\pm M,n\pm N}$ from the DFT of the HR image. For shifts of (0,0), (1,0), (0,1) and (1,1), this is simply:

$$\begin{pmatrix} G^{0,0}_{m,n} \\ \exp\left(\frac{j\pi m}{M}\right) G^{0,1}_{m,n} \\ \exp\left(\frac{j\pi n}{N}\right) G^{1,0}_{m,n} \\ \exp\left(j\pi\left(\frac{m}{M}+\frac{n}{N}\right)\right) G^{1,1}_{m,n} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} F^{0,0}_{m,n} \\ F^{0,0}_{m+M,n} \\ F^{0,0}_{m,n+N} \\ F^{0,0}_{m+M,n+N} \end{pmatrix} \tag{7}$$

$$\begin{pmatrix} F^{0,0}_{m,n} \\ F^{0,0}_{m+M,n} \\ F^{0,0}_{m,n+N} \\ F^{0,0}_{m+M,n+N} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} G^{0,0}_{m,n} \\ \exp\left(\frac{j\pi m}{M}\right) G^{0,1}_{m,n} \\ \exp\left(\frac{j\pi n}{N}\right) G^{1,0}_{m,n} \\ \exp\left(j\pi\left(\frac{m}{M}+\frac{n}{N}\right)\right) G^{1,1}_{m,n} \end{pmatrix} \tag{8}$$

If you take 4 simulated LR observations from Program 2, with S=2 and shifts as in Eqn. 8, it is in this way possible to compute the frequency response of the HR image from the aliased frequency samples of the LR image. We're not going to provide to code to do this here, because as it turns out, there's a much faster way of doing this that we'll describe in the next section. But if you try this on your own, you can get a SR estimation like our result in Fig. 5.

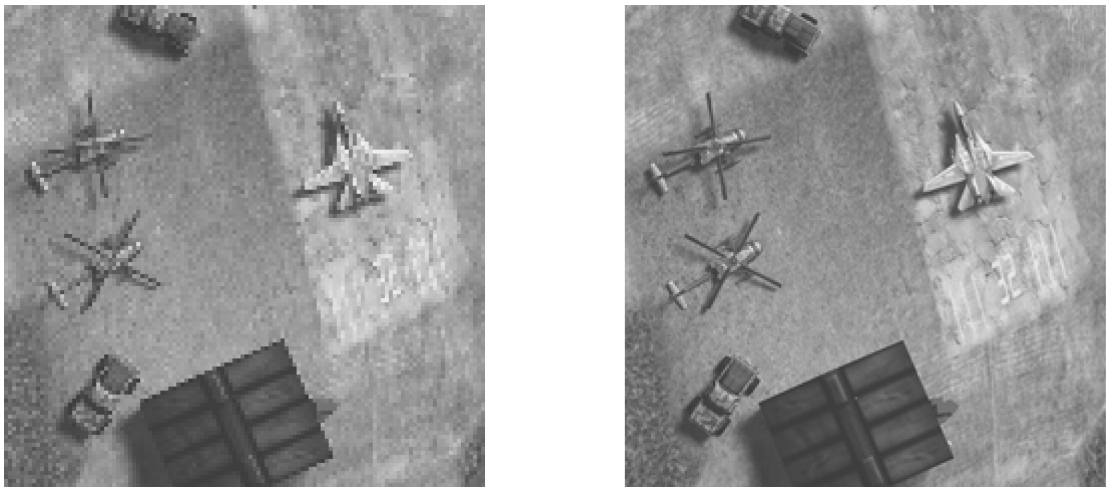

Figure 5. One of four simulated LR observations (left), and the SR estimation (right) using a repeated application of the frequency-based approach in Eqn. 8.

It's useful to take a moment and consider the (maybe counterintuitive) importance of aliasing in SR: SR would be impossible without it.[7] With aliasing, we know that each LR image contains in it information about the high-frequency content of the desired HR image. Granted, this information is mixed in with information about the low-frequency content, but with enough LR images, we can hopefully figure out which part is which, and we get our SR estimation.

But, suppose aliasing did not occur. This would mean one of two things: either that the desired HR image contains only relatively low-frequency content, or that the blur function $\mathbf{H}_i$ removed all of the high-frequency content before the LR image was downsampled. In the first case, we don't need SR, because any given LR image would contain all of the relevant frequency information about the scene. In the second case, we run into a bit of a problem, because it means that we will theoretically never recover that information again. If as an extreme example, we consider a blur function $\mathbf{H}_i$ which only returns the average value (DC component) of the image, then each LR image will be a noisy observation of that single value.

Now that we've discussed the theoretical foundation of super-resolution in the frequency domain, it will be useful in the next section to return to looking at the problem in the spatial domain, where we will find an incredibly simple alternative solution to the problem that we solved in Fig. 5.

3. SHIFT-ADD LR FUSION

Even though the theoretical basis for SR is best explained in the frequency domain, many SR estimation approaches take place in the spatial domain.[8] In particular, the spatial domain allows for a fairly intuitive reconstruction approach called "shift-add fusion", which can be a crucial initialization step in more elaborate SR estimation algorithms. The SR estimation approaches discussed throughout the remainder of this chapter will build upon shift-add fusion.

Shift-add fusion is based on a simple fact: if we know the motion estimates of a set of LR observations (either a priori, or through a separate estimation procedure), then we know the locations on the HR grid that every LR observation pixel "came from." Basically, the downsampling operator $\mathbf{D}_i$ is invertible, and because we assume that we know the motion estimates $\mathbf{F}_i$, we know that the combined motion compensation/downsampling operator $\mathbf{D}_i\mathbf{F}_i$ maps each LR observation pixel from one and only one location on the HR image plane (Fig. 6).

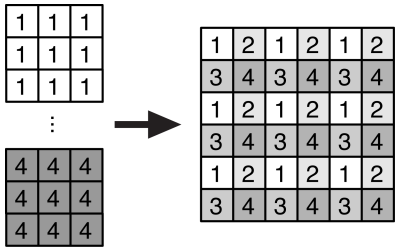

Figure 6. Under known motion estimates, every pixel in the LR observations is known to come from a specific location on the HR image plane.

Shift-add fusion begins with a blank image at the resolution of the unknown HR image. We then cycle through all the LR observations, and "place" every LR observation onto the HR plane, essentially undoing the effects of camera-subject motion and downsampling. If there is negligible blur and noise, this single step can produce a complete SR estimation.[9,10]

Program 3 demonstrates how to perform shift-add fusion with a complete set of LR observations (4 shifted observations for a magnification S of 2), with no blur or noise.

Program 3 Shift-add fusion in spatial domain with 4 LR aliased images, known integer-valued motion estimates.

```
S = 2;
NImages = 4;
dx = {0, 0, 1, 1};
dy = {0, 1, 0, 1};
NoiseStd = 0;
K = {{1}};

(* Simulate LR Observations *)
(* insert Program 2 here *)

(* Perform Shift-Add Fusion *)
SRImage = ConstantArray[0, {H, W}];
For[ImCount = 1, ImCount <= NImages, ImCount++,
  PixelData = ImageData[ImageReflect[SetOfImages[[ImCount]]]];
  For[i = 0, i < H/S, i++,
    For[j = 0, j < W/S, j++,
      px = j*S + dx[[ImCount]];
      py = i*S + dy[[ImCount]];
      SRImage[[py + 1, px + 1]] = PixelData[[i + 1, j + 1]];
    ];
  ];
]

(* Compare the ground-truth and reconstructed images *)
Show[InputImage]
Show[ImageReflect[Image[SRImage]]]
```

An example result of shift-add fusion is shown in Fig. 7. And the reader might be wondering at this point if we're trying to pull a fast one: the result here seems identical to the result in Fig. 5, but as it turns out, the results are identical up to rounding error. The difference lies only in the approach: with the shift-add approach, all we have to do is move pixels from a set of small pictures onto corresponding points in a large picture, whereas before we had to compute 4 DFTs, compute 4 complex multiplications and additions per element in the HR image, and then compute the IDFT of the HR frequency response. Clearly, the spatial-domain approach has a benefit here.

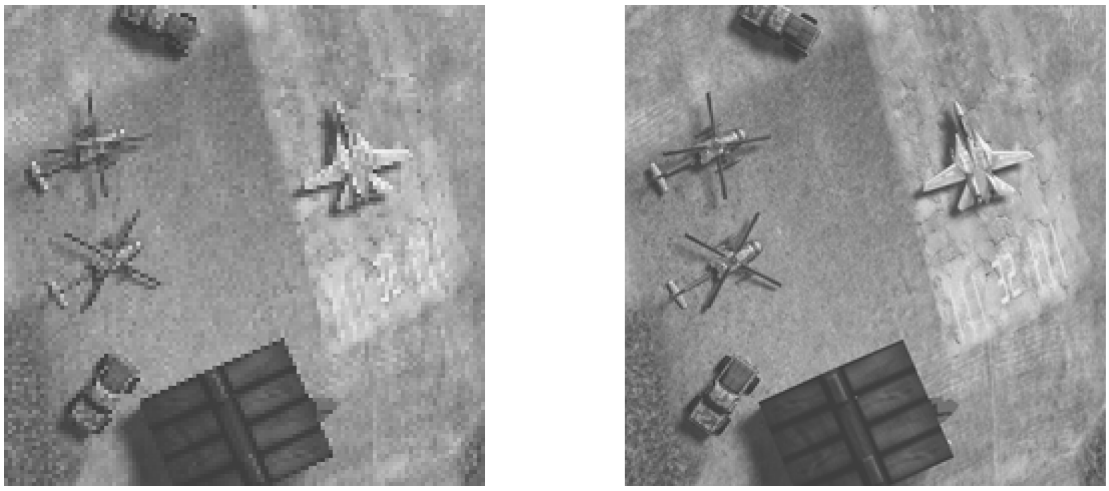

Figure 7. One of four simulated LR observations (left), and the SR estimation using Program 3 (right).

As we said before, shift-add fusion is a typical initialization step in SR algorithms, once motion estimates are known (more on this last caveat is Section 7). However, we will never actually be faced with the problem in Fig. 7, where all of our measurements are perfectly reliable (no noise), each LR pixel is influenced by a single HR pixel (no blur), and every HR pixel was recorded as a LR pixel in one of the observations (complete set of measurements). To deal with the realistic case where our observation set has shortcomings that keep shift-add fusion from being a perfect one-step solution, we in the next section discuss methods of imposing prior knowledge on what we expect the final SR estimate to look like.

4. IMAGE REGULARIZATION

Unfortunately, the set of observed LR images may have some serious shortcomings. It's possible that there are insufficient LR images to completely fill the HR image through shift-add fusion (e.g. S = 2, but NImages < 4). It's also possible that there is sufficient noise in the LR images that even if there are no holes in the fused image, we know that the fused image only represents a noisy version of the true HR image (i.e. the data is complete, but unreliable). To deal with both of these effects, it is standard to employ image regularization in an SR reconstruction algorithm.

Image regularization is motivated by the fact that we might have a good sense of what a "correct" image looks like, and that we might be able to formulate this sense of correct-ness into a mathematical function of the image. Suppose that this function is referred to as $\mathcal{V}(\cdot)$, and the output of $\mathcal{V}(\hat{\mathbf{Z}})$ is a single non-negative score, which is small when the image estimate $\hat{\mathbf{Z}}$ appears like we expect an image to appear, and which is large when $\hat{\mathbf{Z}}$ appears to contain artifacts or noise. For a well-chosen regularization model, we would expect, for example, that the image on the left in Fig. 8 would have a lower corresponding penalty score than the image on the right. Common assumptions in regularization models are smooth gradients, sharply-defined edges, and low pixel variation in otherwise smooth areas.[11]

Figure 8. A comparison between an image which appears "correct" (left) and one which contains notable artifacts (right).

In a regularized reconstruction approach, there is also a fidelity function $\mathcal{F}(\cdot)$, which returns a single non-negative score that describes how closely the current reconstruction estimate matches the observations. In super-resolution, the nearly universally-chosen fidelity function is the sum-squared error between the observations, and what the observations *should* be, given the current SR estimate $\hat{\mathbf{Z}}$. If N is the number of LR observations (indexed by n), then this fidelity function $\mathcal{F}_{SSE}(\cdot)$ can be expressed as:

$$\mathcal{F}_{SSE}(\hat{\mathbf{Z}}) = \sum_{n=1}^{N} \left\| \mathbf{y}_n - \mathbf{D}_n \mathbf{F}_n \mathbf{H}_n \hat{\mathbf{Z}} \right\|^2 \tag{9}$$

A low score from $\mathcal{F}(\hat{\mathbf{Z}})$ indicates that for the estimate $\hat{\mathbf{Z}}$, the predicted LR images closely match the observed LR images, and a high score indicates that the predicted LR images are not close to the observed LR images.

Given the fidelity function $\mathcal{F}(\cdot)$ and the regularization model $\mathcal{V}(\cdot)$, regularized SR estimation can be formulated as an optimization problem: find the SR estimate $\hat{\mathbf{Z}}$ which minimizes $J(\cdot)$, a weighted sum of the two penalty functions (Eqn. 10).

$$J(\hat{\mathbf{Z}}) = \mathcal{F}(\hat{\mathbf{Z}}) + \lambda \mathcal{V}(\hat{\mathbf{Z}}) \tag{10}$$

The tuning parameter λ is essentially used to control how much the algorithm trusts the observations. If λ is small, then the observed LR images are considered reliable, and the fidelity penalty dominates $J(\cdot)$. If λ is larger, the SR estimate will be steered to a larger degree by the regularization model. In the literature, some approaches require λ to be chosen by a user (where the user adjusts λ until the result looks optimal), while some more sophisticated approaches choose λ based on an estimation of the amount of noise in the LR observations.

An important consideration in choosing a regularization model is the ease (or difficulty) with which an optimization algorithm can find a minimum of the objective function J. An easy trap to fall into would be to choose an incredibly detailed regularization model which is difficult to optimize, and for which poor optimization leads to worse results than a good optimization of a simpler regularization model. For this reason, many regularization models in the literature also display nice optimization properties, such as convexity or continuous derivatives.

Image regularization is admittedly ad-hoc as we've laid it out above: it's a practical solution to improving estimation performance, but there is no strong motivation for the approach from theory. Suppose, however, that $\mathcal{F}$ is chosen as the negative log-likelihood of the LR observations given the candidate $\hat{\mathbf{Z}}$ (under the assumption of AWGN, this choice is equivalent to a scaled version of $\mathcal{F}_{SSE}$ from Eqn. 9). Suppose also that $\mathcal{V}$ is chosen as the

negative log-likelihood of the prior probability distribution on $\mathbf{Z}$ (assuming that a probability distribution exists for an image). Under this pair of choices for the penalty functions, then the regularized SR image estimation approach becomes the Bayesian maximum a-posteriori (MAP) estimation of the true SR image.[12]

We advise caution here, though: accurately claiming to have the optimal MAP estimate requires the accurate definition of the prior distribution on $\mathbf{Z}$, which in general is no simple thing to define.[13] In some specific SR applications (e.g. images of an unknown object which is known to be from a set of several known objects), quantifiable prior knowledge is known of the desired HR image, but this is not true for all SR applications. Generally, the definition of a prior distribution is guided by the same sort of intuition that guided the choice of a regularization model, and the MAP approach is no more certainly optimal than the ad-hoc image regularization approach with a well-chosen regularization model.

Program 4 is an example of regularized SR estimation: we assume a magnification of 3, no blurring or noise, and known integer motion estimates, but now we only have 7 of the 9 LR observations. Shift-add fusion results in 22% of the HR image still undetermined, and the regularization model is relied on to fill in the holes. This specific type of regularized image reconstruction (filling in missing pixels) is alternatively referred to as "inpainting", and has well-established approaches in the literature.[14] Inpainting is also part of the fixed functionality of Mathematica.

The regularization model chosen in Program 4 is the total-variation (TV) model. TV regularization applies an absolute value penalty to differences in the values of neighboring pixels, and has been observed to handle sharp edges well (i.e. sharp edges are not unrealistically penalized).[15,16]

Program 4 Shift-add fusion in spatial domain with 7 LR aliased images, and TV inpainting.

```
S = 3;
NImages = 7;
dx = {0, 2, 1, 1, 2, 0, 2};
dy = {0, 0, 1, 2, 1, 2, 2};
NoiseStd = 0;
K = {{1}};

(* Simulate LR Observations *)
(* insert Program 2 here *)

(* Perform Shift-Add Fusion *)
FusionImage = ConstantArray[0, {H, W}];
FlagImage = ConstantArray[0, {H, W}];
For[ImCount = 1, ImCount <= NImages, ImCount++,
  PixelData = ImageData[ImageReflect[SetOfImages[[ImCount]]]];
  For[i = 0, i < H/S, i++,
    For[j = 0, j < W/S, j++,
      px = j*S + dx[[ImCount]];
      py = i*S + dy[[ImCount]];
      FusionImage[[py + 1, px + 1]] = PixelData[[i + 1, j + 1]];
      FlagImage[[py + 1, px + 1]] = 1;
    ];
  ];
]

(* Fill in holes in the fused image *)
SRImage = Inpaint[Image[FusionImage], Binarize[Image[FlagImage], {0,0}],
            Method -> {"TotalVariation"}];

(* Compare the ground-truth and reconstructed images *)
Show[InputImage]
Show[ImageReflect[SRImage]]
```

Looking at the output of Program 4 while varying NImages, we can see the effect of decreasing the number of LR observations (Fig. 9). As the number of LR images decreases (and the number of holes in the fused image increases), the estimation has to rely more heavily on the regularization model.[17] And as we might intuitively have guessed, the quality of the estimation decreases as the number of LR images decreases, but the important feature to notice here is at what rate the result becomes unacceptable. The estimation from 7 observations, for example, is almost indistinguishable from the estimation from all 9 observations (which requires no inpainting). Even the estimation from 5 LR observations is still acceptable, despite the fact that the corresponding fused image was missing almost 50% of the data. The important trend that this illustrates is that with a well-chosen regularization model, the SR estimation can still be good, even if the observed dataset is less than perfect.

Figure 9. The result of regularized SR estimation, for S= 3. Top row: results using (left to right) 9, 7, or 5 LR observations. Bottom row: results using (left, center) 3 or 2 LR observations, and an example LR observation (right).

In the example above, the regularization model was used to estimate missing pixels in the fused HR image, but regularization can also help with more subtle shortcomings of the observed data. In the next section, we will address the case where each LR observation suffers from blur and noise. In this case, instead of using regularization to fill in blatant holes in the data, we will use it to fill in less obvious ambiguities, steering the SR estimation to a good solution in the face of unreliable observations.

5. SR RECONSTRUCTION IN THE PRESENCE OF BLUR

In Section 2, we said that undersampling in LR observations is what makes super-resolution possible. However, the high-frequency content of the HR image can still be attenuated by blur before being observed as aliasing in the LR image. This section deals with SR reconstruction from a set of LR images suffering from blur.

Image blur is typically modeled by a blur kernel, sometimes also referred to as a point-spread function (PSF). The PSF K describes how a point of light in an unblurred image I is redistributed in a local neighborhood in the blurred observation O of that image:[18]

$$O(x,y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} I(x+i, x+j) \mathcal{K}_{x,y}(i,j) \tag{11}$$

In this most general version of the blurring model, the PSF can vary with the location of the ideal image point (x, y), but it is a common assumption that the PSF is spatially-invariant across the image plane. Under this assumption, Eqn. 11 can be expressed as the convolution of the unblurred image with the PSF to produce the blurred image. Constraints placed on the PSF by definition are that every element is non-negative, and that the elements sum to unity.

In the observation of LR images, blurring can occur for a number of reasons. Motion blur is caused by the movement of the subject relative to the camera during the image exposure[19] (and because we typically rely on at least small movement between consecutive images to compute SR, we commonly encounter motion blur in application). Lens blur can be due to aberrations (imperfections in the design or manufacture of the lens), or due to defocus (the subject of the image is either farther or closer than the focus distance of the camera).[20] Atmospheric blur is common in remote-sensing applications (like satellite imaging), and models the effect of looking through a large volume of air containing turbulence or particulate matter like dust.

The final source of blurring (and the only source that is guaranteed to be measurably present in any SR application) is the spatial integration of light on the imaging sensor. By this, we mean that each pixel on the imaging sensor is of finite dimensions, and that all light falling across that pixel area will be recorded as having come from that single location.[21] This is well modeled by considering a HR image (which has already been motion-compensated, and affected by all other sources of blur), and blurring that HR image with a 2D box filter of the dimensions of the pixel (e.g. for an SR magnification of 3 and assuming 100% fill factor, this would be a 3x3 uniform blur kernel applied to the HR image).[22] After this final source of blurring is applied, downsampling of the HR image and addition of noise produce the LR observation.

In addition to the assumption of blur that is spatially-invariant across the image plane, it's also common to assume that the blur is constant across the set of LR images. For lens and atmospheric blur, this assumption can hold well for image sets where the camera-subject distance doesn't vary significantly across images. Motion blur due to jitter (small random movements in either the camera or subject) can easily violate this assumption,[23] but large movement (e.g. a stationary camera and a car passing at a constant rate) can still match this assumption well. Finally, the blur due to the sensor can be well-modeled as invariant across the set of LR images if there is not significant rotation, scale, or perspective change across the set of LR images (as any of these would change the shape of the 2D box filter relative to each LR image).[24]

In the presence of blur which is spatially invariant across the image plane, and which is consistent across all LR observations, we can restructure the SR problem in an interesting way: the blur operator $\mathbf{H}$ operates identically on the HR image $\mathbf{Z}$ for all LR observations, and the two terms can be grouped together:

$$\mathbf{y}_i = \mathbf{D}_i\mathbf{F}_i\left(\mathbf{HZ}\right) + \mathbf{n}_i \tag{12}$$

As we discussed in Section 3, shift-add fusion is an attempt to undo the combined effect of the downsampling and motion operators ($\mathbf{D}_i\mathbf{F}_i$). So, we see in Eqn. 12 that if we perform shift-add fusion on a set of LR observations which are all affected by uniform blur $\mathbf{H}$, then we get back a noisy observation of the blurred HR image ($\mathbf{HZ}$).

The reason that this restructuring is useful is that given a noisy observation of a blurred image, and knowing the blur kernel K (which is equivalently expressed[†] in the blur operator $\mathbf{H}$), there are well-established methods to extract out an estimate of the unblurred image $\hat{\mathbf{Z}}$.[25] These deblurring methods are referred to broadly as "deconvolution," and they typically involve imposing a regularization model on the unblurred image estimate, just as we did in Section 4.

In Program 5, we take this two-stage approach to computing a SR reconstruction from LR observations suffering from noise and blur. Shift-add fusion is performed as before (and because there are no holes in the fused data, inpainting is not necessary), and deconvolution is performed using Mathematica's fixed functionality. The regularization method for the deconvolution is chosen to be total variation (TV), just as it was with the inpainting in Program 4.

[†]There is a reason for this seemingly redundant notation: the blur kernel K is defined in the spatial domain of the image, where it has a clear interpretation from a computer-science perspective, whereas the blur operator $\mathbf{H}$ is a matrix that operates on the image organized lexicographically as a vector, where it has a clear interpretation from a mathematics perspective. In practice, the blur operator $\mathbf{H}$ is only used in looking at the mathematical implications of an approach, and the blur kernel K is actually used in implementation.

Program 5 Two-stage SR with LR observations suffering from uniform blur and AWGN.

```
S = 3;
NImages = 9;
dx = {0, 0, 0, 1, 1, 1, 2, 2, 2};
dy = {0, 1, 2, 0, 1, 2, 0, 1, 2};
NoiseStd = 2/255;
K = GaussianMatrix[2.5];

(* Simulate LR Observations *)
(* insert Program 2 here *)

(* Perform Shift-Add Fusion *)
FusionImage = ConstantArray[0, {H, W}];
For[ImCount = 1, ImCount <= NImages, ImCount++,
  PixelData = ImageData[ImageReflect[SetOfImages[[ImCount]]]];
  For[i = 0, i < H/S, i++,
    For[j = 0, j < W/S, j++,
      px = j*S + dx[[ImCount]];
      py = i*S + dy[[ImCount]];
      FusionImage[[py + 1, px + 1]] = PixelData[[i + 1, j + 1]];
    ];
  ];
]

(* Deblur the fused HR Image with TV-regularized deconvolution *)
SRImage = ImageDeconvolve[ Image[FusionImage], K,
            Method -> "TotalVariation"];

(* Compare the ground-truth and reconstructed images *)
Show[InputImage]
Show[ImageReflect[SRImage]]
```

Looking at the results (Fig. 10), we see that as the strength of the noise increases, the quality of the SR estimation decreases (as we would expect). The interesting aspect here is to see how the regularization model "reacts" to noisy input: the model tries to avoid estimation results that contain noise or ringing effects near edges (sometimes caused by strictly deconvolving a noisy blurred image in the frequency domain[26]), so the regularization model tends to increasingly segment the output into regions that maintain sharp edges. The downside of this is the rapid loss of texture in the output (where texture is easily confused for the effect of noise), but even at high noise levels, the silhouette and average tone of objects is maintained in the output.

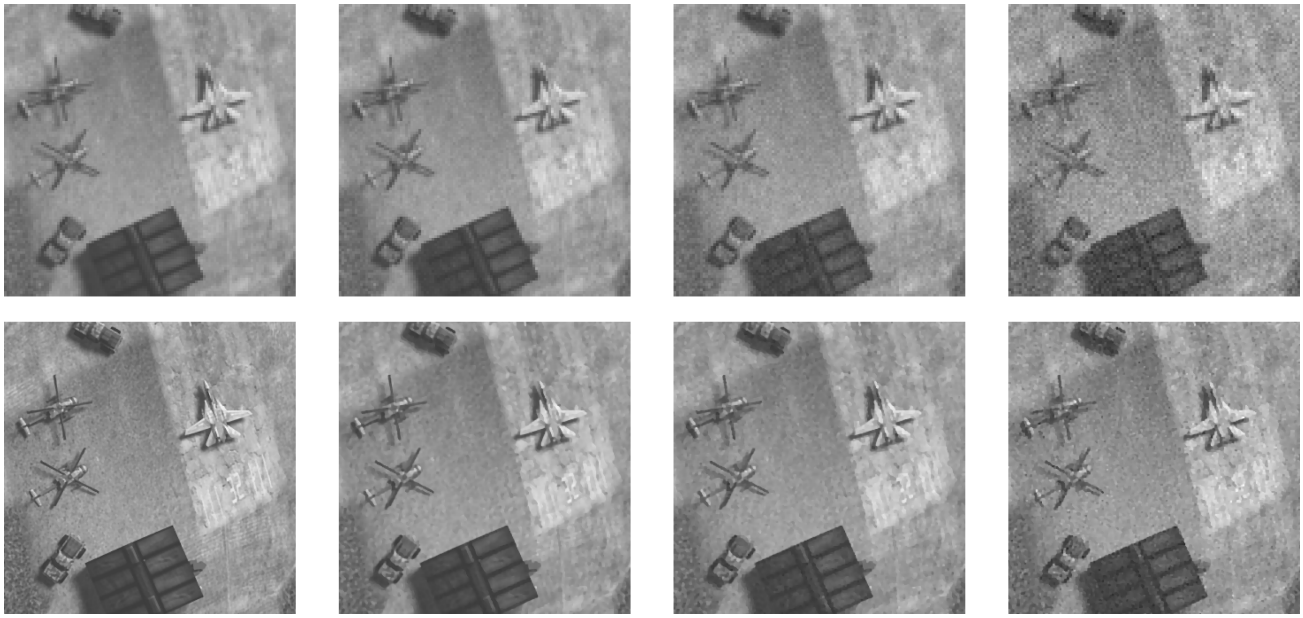

Figure 10. The effect of noise strength on SR reconstruction in the presence of blur. Top row is LR observation, bottom row is SR result with TV regularization. NoiseStd is chosen as (from left to right), 0/255, 2/255, 5/255, and 10/255.

It might be tempting to combine Programs 4 and 5, in order to attack the SR problem where there is noise, blur, and an incomplete number of observations. We encourage the reader to try this, but we stress that a quick combination of these two programs will not yield an optimal solution. Each one of these approaches (faced with either holes in data, or unreliable data, but not both) was able to take advantage of the fixed functionality in Mathematica, but a more general approach cannot do this. In practice, SR reconstruction is performed iteratively, where the algorithm may begin with shift-add fusion, but then an iterative optimization approach is taken to minimize the objective function $J(\cdot)$ from Section 4 for a given choice of regularization model.[8] Inpainting and deconvolution can both play important roles in this optimization (especially in determining a good initial estimate of $\hat{\mathbf{Z}}$), but a more involved optimization algorithm is required for this more involved problem.

6. SR WITH COLOR IMAGES

Up to this point, we've only dealt with monochrome images, but super-resolution is also possible with color images. This problem is fundamentally more difficult, but it's becoming a more typical application of super-resolution algorithms.

The formation of color LR observations is nearly identical to the formation of monochrome LR observations, with one significant difference: the introduction of the color filter array (CFA). The CFA is how nearly every[‡] digital imaging system handles color, by placing a color filter in front of each pixel on the sensor and restricting that pixel to record information for a single color channel (red, green, or blue).[27] In this chapter, we restrict ourselves to the ubiquitous choice of the Bayer mask as the CFA, shown below.

[‡]Except for three-chip cameras, which use a prism and three sensors, each dedicated to a separate color channel. This approach increases color fidelity, but also significantly increases the cost of the system. The vast majority of digital color imaging systems utilize a CFA, so that's what our treatment here will deal with.

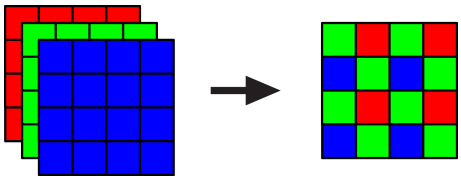

Figure 11. The observation of a full-color image (left), and the resulting mosaiced image (right) resulting from the Bayer CFA.

This recorded image is referred to as a "mosaiced" color image, and before being shown to the user in a consumer application, this mosaced image is run through a demosaicing algorithm which interpolates the missing color information for each pixel. Because a raw mosaiced image is unpleasant to view, demosaicing algorithms are essential to consumer imaging applications, but demosaicing presents a problem from an analysis standpoint: it involves essentially "guessing" 67% of the information in the full-color image. Sophisticated demosaicing algorithms can do a very effective job of this guessing, but to avoid artifacts in our SR reconstruction, we restrict ourselves to the recorded mosaiced image.

In the LR observation model for a color image, the CFA can be accounted for simply by changing the downsampling pattern $\mathbf{D}$ across the color channels. Rather than sampling every 2nd HR pixel, for example, we can model the green channel as sampling every 4th HR pixel with an offset of 2 between rows, and the red and blue channels as sampling every 4th HR pixel, and alternating rows.[28]

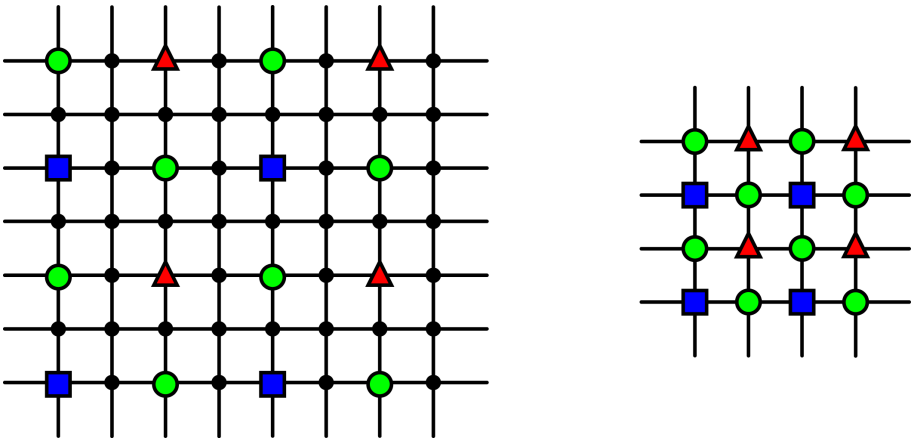

Figure 12. The downsampling grid of a CFA color image on the HR image plane with a magnification of 2 (left), and the resulting mosaiced color LR image (right).

If the color channel is indexed by c, the observation of each color channel $\mathbf{y}_{i,c}$ in the LR observations is modeled as:

$$\mathbf{y}_{i,c} = \mathbf{D}_{i,c}\mathbf{H}_{i,c}\mathbf{F}_i\mathbf{Z}_c + \mathbf{n}_{i,c} \tag{13}$$

We notice that the motion estimate $\mathbf{F}$ still depends only on the index i of the LR image, and not on the color channel c. The effect of blurring, $\mathbf{H}$, can depend on the color channel, and some work in the literature has demonstrated that for spatially-varying PSFs (spatially varying across the image plane), chromatic aberrations can be accounted for by allowing the PSF to vary slightly across the color channels.[29] Typically, however, the blur kernels for each of channel, K_R, K_G, K_B, are modeled as identical, and are example below will also make that assumption.

As a Mathematica example, we simulate a set of LR observation images from a HR ground-truth image (which is RGB color), and we allow the definition of the CFA for the simulated camera. The CFA is defined as

a 2x2 matrix, where each element is the index of the color channel to which that CFA element is sensitive: the example below is the typical Bayer mask, where the top row is sensitive to {Green, Red}, and the bottom row is sensitive to {Blue, Green}. We also allow for blurring in the image, but the blur kernel K is defined as a delta function in the example below.

Program 6 Simulation of LR Observations with a CFA

```
S = 2;
NImages = 4;
dx = {0, 0, 1, 1};
dy = {0, 1, 0, 1};
SetOfImages = Array[0, {NImages}];
K = {{1}};

(* Define the CFA *)
CFA = {{2, 1}, {3, 2}};

(* Import a color HR image, and blur it channel-wise *)
InputImage = ColorConvert[Import["TestImage.jpg"], "RGB"];
ImDim = ImageDimensions[InputImage];
W = ImDim[[1]] - Mod[ImDim[[1]], S];
H = ImDim[[2]] - Mod[ImDim[[2]], S];
InputImage = ImageCrop[InputImage, {W, H}];
BlurIm = ImageConvolve[InputImage, K];

(* Simulate the set of LR observations with the CFA *)
For[ImCount = 1, ImCount <= NImages, ImCount++,
  SmallImArray = ConstantArray[0, {H/S, W/S, 3}];
  For[i = 0, i < H/S, i++,
    For[j = 0, j < W/S, j++,
      px = j*S + dx[[ImCount]] + 0.5;
      py = i*S + dy[[ImCount]] + 0.5;
      ColorVal = ImageValue[InputImage, {px, py},
              Resampling -> "Bilinear",
              Padding -> "Fixed"]
              + NoiseStd*RandomVariate[NormalDistribution[0,1]];
      CFAChannel = CFA[[Mod[i, 2] + 1, Mod[j, 2] + 1]];
      SmallImArray[[i+1, j+1, CFAChannel]] = ColorVal[[CFAChannel]];
    ];
  ];
  SetOfImages[[ImCount]] = ImageReflect[Image[SmallImArray]];
]
```

If we look at these new simulated LR observations, we see that each LR pixel contains information about only one color channel, and that there is twice as much information about the green channel as the red or blue channels.

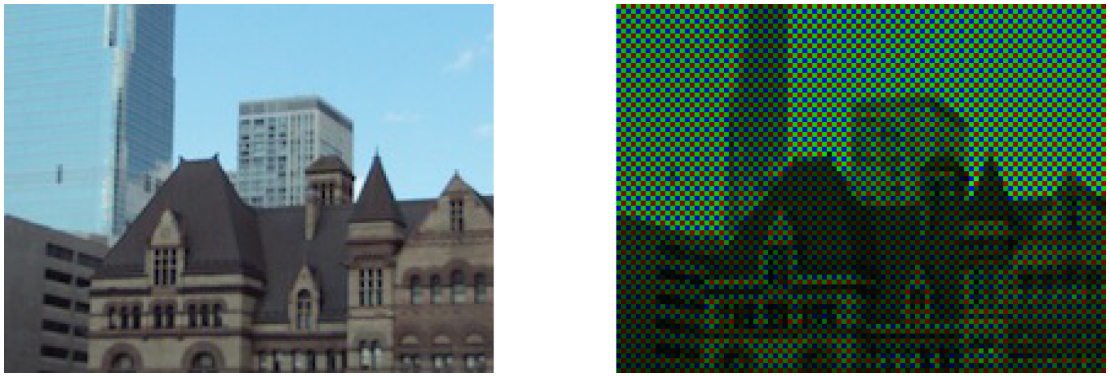

Figure 13. A cropped section of a full-color image (left), and a corresponding mosaiced LR observation (right).

Given this new set of LR observations, we now face the problem of reconstruction. Shift-add fusion is a bit more interesting now, because each observed pixel in a LR image corresponds to a sample of a single color at a single location on the HR plane. This means that in the fused HR image, we might have no color information for a given HR pixel, we might have information about only a single color channel, or we might have information about multiple color channels, depending on the motion estimates.[1]

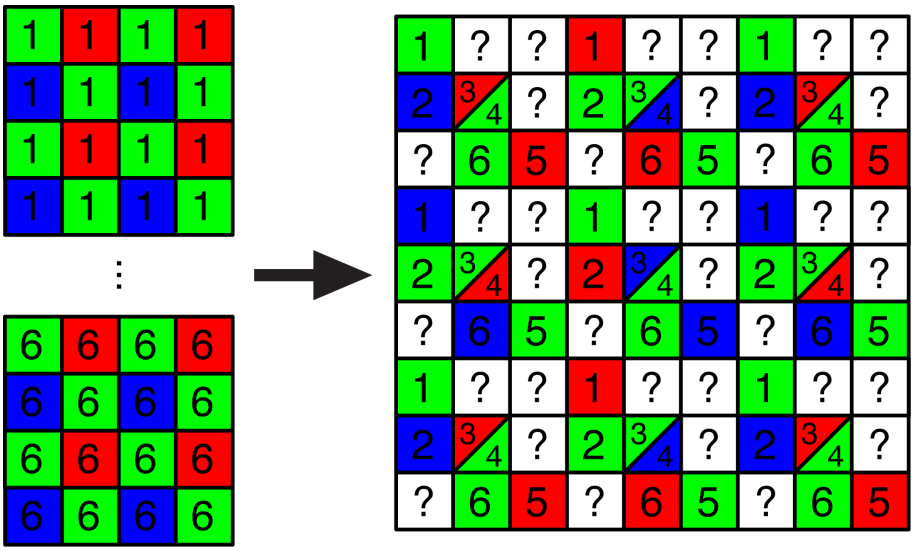

Figure 14. Shift-add fusion with mosaiced LR images.

One approach to SR reconstruction is to fill in the holes in the fused data with TV inpainting for each color channel, and then recombine the three separate color channel SR reconstructions into one color image. This approach (given in the program below) is similar to performing the reconstruction approach from Program 4 separately on each color channel, and taking the CFA into account when performing shift-add fusion.

Program 7 Shift-Add Fusion and TV Inpainting for CFA Observation Images

```
(* Perform Shift-Add Fusion with LR CFA Observations *)
FusionImage = ConstantArray[0, H, W, 3];
FlagImage = ConstantArray[0, H, W, 3];
For[ImCount = 1, ImCount <= NImages, ImCount++,
  PixelData = ImageData[ImageReflect[SetOfImages[[ImCount]]]];
  For[i = 0, i < H/S, i++,
    For[j = 0, j < W/S, j++,
      px = j*S + dx[[ImCount]];
      py = i*S + dy[[ImCount]];
      CFAChannel = CFA[[Mod[i, 2] + 1, Mod[j, 2] + 1]];
      FlagImage[[py+1, px+1, CFAChannel]] = 1;
      FusionImage[[py+1, px+1, CFAChannel]] =
              PixelData[[i+1, j+1, CFAChannel]];
    ];
  ];
]

(* Perform Inpainting Separately on each color channel *)
FusionChannels = ColorSeparate[Image[FusionImage]];
FlagChannels = ColorSeparate[Image[FlagImage]];
For[ChannelCount = 1, ChannelCount <= 3, ChannelCount++,
  FusionChannels[[ChannelCount]] =
    Inpaint[FusionChannels[[ChannelCount]],
            FlagChannels[[ChannelCount]],
            Method -> {"TotalVariation"}];
];
SRImage = ImageReflect[ColorCombine[FusionChannels, "RGB"]];

(* Compare the ground-truth and reconstructed images *)
Show[InputImage]
Show[SRImage]
```

If we look at the results of this approach, we see that the performance does not seem nearly as good as it was with monochrome images. In particular, we notice color artifacts at sharp edges, and poor representation of areas containing high-frequency information. These shortcomings underscore what we said at the beginning of this section: super-resolution with color images is a fundamentally more difficult problem.[30]

Figure 15. The HR ground-truth (left) versus the SR reconstruction (right).

In a broad sense, super-resolution with color images is more difficult because we face a worse ratio of input information to output information. The introduction of the CFA means that with 4 LR observations, we're really only getting 2 observations worth of information in the green channel, and only 1 observation worth of information in the red and blue channelss. It's left as an exercise for the reader to modify the example code to observe 16 LR images, with carefully chosen motion estimates ranging from 0 to 3 in each axis, to get back a lossless reconstruction of the full-color image. We also encourage the reader to observe the change in results as the blur kernel K is modified to result in less severely undersampled images.

We also note that the above reconstruction approach uses a monochrome regularization model that works spatially, but not across color channels: each color channel is inpainted without regard to the content of the other color channels. This unfortunately ignores inter-channel information that can benefit reconstruction.[31] Sharp edges, for example, are commonly assumed to occur in the same location across all three channels, and this assumption helps avoid the color artifacts at sharp edges like those in Fig. 15. Other assumptions by sophisticated color regularization models include imposing that luminance fidelity has priority over chrominance fidelity, and imposing an isotropic smoothness-enforcing penalty equally across both chrominance channels.[32]

7. MOTION ESTIMATION

The reader might notice, when considering a real-world application of the SR approaches discussed so far, that we've made a pretty unrealistic assumption: that we know, automatically, what the motion estimates are for the LR observations. In application, it would be much more realistic to begin the problem with only the set of LR observations, and no additional information. In this more realistic case, it's necessary to compute motion estimates directly from the content of the LR observations.

Motion estimation approaches in the literature can generally be separated into two categories: area-based approaches (where the estimator uses all the pixel data in the image) and feature-based approaches (where the estimator identifies point features in the images, and then only deals with the locations and descriptions of those features).

Two popular area-based approaches are block matching and optical flow.[33] Block matching attempts to match small windows of one image to corresponding small windows in another image.[34] Each matched block gives a local translation for the block from one image to the next, and these local translations can then be used as-measured (which allows for non-global translational motion), or can be used to solve for global motion which includes translation as well as optionally scale and rotation. Optical flow methods estimate local motion between consecutive images by taking advantage of temporal correlation across the LR observation set (assuming that the LR images are temporally organized and are effectively or actually frames from a video sequence).[35] Similar to block matching, optical flow produces a set of local translation estimates across the image plane that can be used to represent non-global translational motion, or can be consolidated into a global affine transformation estimation.

Feature-based approaches, which borrow from work elsewhere in computer vision, compute motion estimates from point correspondences.[36] There are established models in the literature to find "feature points" (points in an image which are easily localized and described[37]), and to match a set of feature points in one image to a corresponding set of feature points in a separate image. Especially when these features points can be identified to sub-pixel accuracy, this set of point correspondences can then be used to compute an accurate motion estimate between that pair of images.

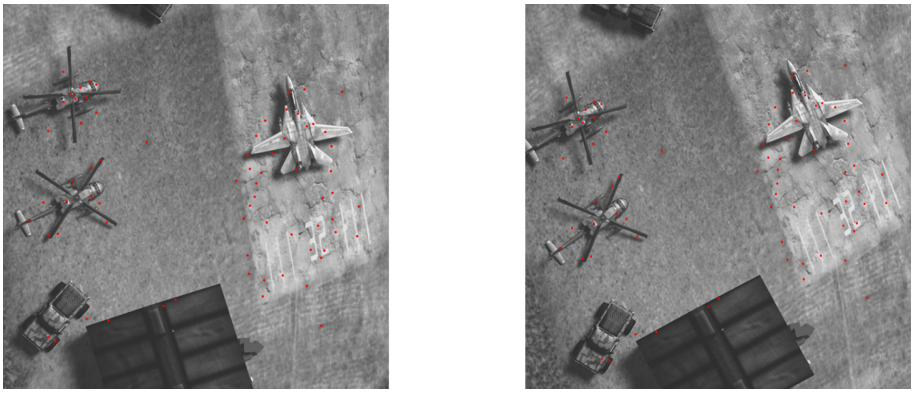

Figure 16. A set of identified feature points that have been matched between a pair of images.

In Program 8, we solve for the global translational motion of a set of LR observations by using a feature-based approach. For every pair of images in the observation set, feature point correspondences are determined using Mathematica's fixed functionality, and a global translation is computed (i.e. the shift from one image in the pair to the other). The linear set of motion equations (where each equation represents the shift estimate between a unique pair of LR observations) is then solved in a least-squares manner for the $NImages - 1$ motion estimates, relative to the first image in the observation set.

Program 8 Compute translational motion estimates from the set of LR observations

```
(* Setup the variables *)
NRows = Binomial[NImages, 2];
TX = ConstantArray[0, NRows, 1];
TY = ConstantArray[0, NRows, 1];
A = ConstantArray[0, NRows, NImages];

(* Form the set of linear equations from every *)
(* pairwise image correspondence in the set *)
RowIndex = 0;
For[i = 1, i < NImages, i++,
  For[j = i + 1, j <= NImages, j++,
    RowIndex += 1;

    (* Compute point correspondences for the image pair *)
    Matches = ImageCorrespondingPoints[
                SetOfImages[[i]], SetOfImages[[j]]];

    (* Compute the translation estimate for the pair *)
    {RowErr, Trans} = FindGeometricTransform[Matches[[1]], Matches[[2]],
                "Transformation" -> "Translation",
                "Method" -> "RANSAC"];
    T = Trans[{0, 0}];

    (* Append the information to the linear equations *)
    A[[RowIndex, j]] = 1;
    A[[RowIndex, i]] = -1;
    TX[[RowIndex]] = T[[1]];
    TY[[RowIndex]] = T[[2]];
  ];
];

(* Solve for the set of motion estimates *)
dXHat = LeastSquares[A, TX];
dYHat = LeastSquares[A, TY];

(* Adjust the motion estimates to be at HR scale, *)
(* and relative to the first LR image*)
dXHat = S*(dXHat - dXHat[[1]]);
dYHat = S*(dYHat - dYHat[[1]]);
```

The set of motion estimates can then be compared for accuracy against the ground truth (e.g. compare dX and dXHat). As the magnification S increases, it becomes more and more difficult for the motion estimates to be accurate on the scale of the HR image. Also, as aliasing increases in the LR observations, the motion estimates can become less accurate as the feature detectors (which are not designed for use with aliased images) have a more difficult time identifying the position of feature points. We encourage the reader to experiment with different parameters for simulating the LR observation set, and observe the effect on the motion estimate accuracy (e.g. look at the effect of more or less blur, or more or fewer images in the observation set).

We also encourage the reader to incorporate Program 8 into previous code examples (i.e. use the motion estimates in reconstruction, rather than the ground-truth motion that we've used up until this point). To

incorporate Program 8 into previous code, the reader will simply need to round the motion estimates dXHat and dYHat (because previous code has assumed integer-valued translational motion), and use the estimates in place of dX and dY. In many cases, the motion estimates will lead to an SR estimation that is comparable in quality to the estimation using the ground-truth motion (Fig. 17).

Figure 17. Ground-truth HR image (top left), one of six LR observations (top right), the SR estimation from known motion (bottom left) and SR estimation using our new motion estimates (bottom right).

As the motion estimates become less accurate, however, the reader will notice that the SR estimation becomes dramatically less accurate.[25] This underscores the importance of accurate motion estimation in super-resolution, and the reason why this very narrow aspect of the problem is the focus of a large fraction of super-resolution research.

REFERENCES

[1] Farsiu, S., Elad, M., and Milanfar, P., "Video-to-video dynamic super-resolution for grayscale and color sequences," *EURASIP Journal on Advances in Signal Processing* (2006).

[2] Keller, S., Lauze, F., and Nielsen, M., "Video super-resolution using simultaneous motion and intensity calculations," *IEEE Transactions on Image Processing* **20**, 1870–1884 (July 2011).

[3] Chung, J., Haber, E., and Nagy, J., "Numerical methods for coupled super-resolution," *Inverse Problems* **22**, 1261–1272 (2006).

[4] Elad, M. and Feuer, A., "Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images," *IEEE Transactions on Image Processing* **6**(12) (1997).

[5] Borman, S., *Topics in Multiframe Superresolution Restoration*, PhD thesis, University of Notre Dame (2004).

[6] Tsai, R. and Huang, T., [*Advances in Computer Vision and Image Processing*], vol. 1, ch. Multiframe Image Restoration and Registration, 317–339, JAI Press Inc. (1984).

[7] Robinson, D., Farsiu, S., and Milanfar, P., "Optimal registration of aliased images using variable projection with applications to super-resolution," *The Computer Journal* **52**(1), 31–42 (2009).

[8] Karam, L., Sadaka, N., Ferzli, R., and Ivanovski, Z., "An efficient selective perceptual-based super-resolution estimator," *IEEE Transactions on Image Processing* **20**(12) (2011).

[9] Molina, R., Vega, M., Abad, J., and Katsaggelos, A. K., "Parameter estimation in bayesian high-resolution image reconstruction with multisensors," *IEEE Transactions on Image Processing* **12**(12), 1655–1667 (2003).

[10] Vandewalle, R., Sbaiz, L., Vetterli, M., and Susstrunk, S., "Super-resolution from highly undersampled images," in [*International Conference on Image Processing*], **1-5**, 701–704 (2005).

[11] Liu, F., Wang, J., Zhu, S., Gleicher, M., and Gong, Y., "Visual-quality optimizing super resolution," *Computer Graphics Forum* **28**, 127–140 (March 2009).

[12] Shultz, R. and Stevenson, R., "A bayesian approach to image expansion for improved definition," in [*IEEE Trans. on Image Processing*], **3**(3), 233–242 (1994).

[13] Weiss, Y. and Freeman, W., "What makes a good model of natural images?," *IEEE Conference on Computer Vision and Pattern Recognition* , 1–8 (June 2007).

[14] Ng, M., Shen, H., Lam, E., and Zhang, L., "A total variation regularization based super-resolution reconstruction algorithm for digital video," *EURASIP Journal on Advances in Signal Processing* (2007).

[15] He, Y., Yap, K., Chen, L., and Chau, L., "A nonlinear least square technique for simultaneous image registration and super-resolution," *IEEE Transactions on Image Processing* **16**(11), 2830–2841 (2007).

[16] Babacan, S., Molina, R., and Katsaggelos, A., "Variational bayesian blind deconvolution using a total variation prior," *IEEE Transactions on Image Processing* **18**, 12–26 (January 2009).

[17] Mateos, J., Molina, R., and Katsaggelos, A., "Bayesian high resolution image reconstruction with incomplete multisensor low resolution systems," in [*IEEE International Conference on Acoustics, Speech, and Signal Processing*], (2003).

[18] Simpkins, J. and Stevenson, R., "Robust grid registration for non-blind psf estimation," in [*IS&T/SPIE Electronic Imaging: Visual Information Processing and Communication III*], (2012).

[19] Shan, Q., Jia, J., and Agarwala, A., "High-quality motion deblurring from a single image," *ACM Transactions on Graphics* **27**, 1–10 (August 2008).

[20] Born, M. and Wolf, E., [*Principles of Optics*], Cambridge University Press (1999).

[21] Holst, G., [*CCD Arrays, Cameras, and Displays*], SPIE Optical Engineering Press (1996).

[22] Baker, S. and Kanade, T., "Limits on super-resolution and how to break them," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**, 1167–1183 (September 2002).

[23] Whyte, O., Sivic, J., Zisserman, A., and Ponce, J., "Non-uniform deblurring for shaken images," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], (2010).

[24] Borman, S. and Stevenson, R., "Linear models for multi-frame super-resolution restoration under non-affine registration and spatially varying PSF," in [*Computational Imaging II*], *Proceedings of the SPIE* **5299**, 234–245 (January 2004).

[25] Protter, M., Elad, M., Takeda, H., and Milanfar, P., "Generalizing the nonlocal-means to super-resolution reconstruction," *IEEE Transactions on Image Processing* **18**(1), 36–51 (2009).

[26] Lagendijk, R., Biemond, J., and Boekee, D., "Regularized iterative image restoration with ringing reduction," *IEEE Transactions on Acoustics, Speech, and Signal Processing* **36**, 1874–1888 (December 1988).

[27] Kimmel, R., "Demosaicing: Image reconstruction from color ccd samples," *IEEE Transactions on Image Processing* **8**, 1221–1228 (September 1999).

[28] Farsiu, S., Elad, M., and Milanfar, P., "Multiframe demosaicing and super-resolution of color images," *IEEE Transactions on Image Processing* **15**(1), 141–159 (2006).

[29] Joshi, N., Szeliski, R., and Kriegman, D., "Psf estimation using sharp edge prediction," *IEEE Conference on Computer Vision and Pattern Recognition* , 1–8 (June 2008).

[30] Sroubek, F., Cristobal, G., and Flusser, J., "A unified approach to superresolution and multichannel blind deconvolution," *IEEE Transactions on Image Processing* **16**, 2322–2332 (September 2007).

[31] Trimeche, M., Paliy, D., Vehvilainen, M., and Katkovnic, V., "Multichannel image deblurring of raw color components," *IS&T/SPIE Electronic Imaging: Computational Imaging III* (March 2005).

[32] Gotoh, T. and Okutomi, M., "Direct super-resolution and registration using raw cfa images," in [*Computer Vision and Pattern Recognition*], (2004).

[33] Barreto, D., Alvarez, L., and Abad, J., "Motion estimation techniques in super-resolution image reconstruction: A performance evaluation," in [*Virtual Observatory: Plate Content Digitization, Archive Mining and Image Sequence Processing*], (2005).

[34] Callico, G., Lopez, S., Sosa, O., Lopez, J., and Sarmiento, R., "Analysis of fast block matching motion estimation algorithms for video super-resolution systems," *IEEE Transactions on Consumer Electronics* **54**, 1430–1438 (August 2008).

[35] Baker, S. and Matthews, I., "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision* **56**(3), 221–255 (2004).

[36] Singh, M., Lu, C., Basu, A., and Mandal, M., "Choice of low resolution sample sets for efficient super-resolution signal reconstruction," *Journal of Visual Communication and Image Representation* **23**(1) (2012).

[37] Mikolajczyk, K. and Schmid, C., "Scale & affine invariant interest point detectors," *International Journal of Computer Vision* **60**, 63–86 (October 2004).