Yusuf Ozuysal
Andrew Wong

Super-resolution

1 Introduction

Super-resolution deals with the construction of high-resolution images using a set of low-resolution images obtained from a scene with subpixel shifts. These low-resolution images are typically obtained from a jittery camera source, such as a camera mounted on a vibrating aircraft, or a slowly-moving subject, such as a few frames of a standing person in front of a surveillance camera. The low-resolution images have small translations and rotations from the high-resolution reference image. The problem consists of constructing a model of linear transformations for each image, and then piecing together the images to form the high-resolution image. Super-resolution image construction has applications in remote sensing and medical imaging, and in scenarios where directly capturing high-resolution images is not feasible.

In this project, we attempt to solve the super-resolution problem with a MAP approach, following Hardie, et. al [1]. We construct a model of the low-resolution images and attempt to learn the high-resolution images. Effects of model parameters are discussed.

2 Model

The initial part of the problem consists of constructing a model relating the high-resolution image to the set of low-resolution images. From a training set of p low-resolution images sized $M_1 \times M_2$, we represent the kth low-resolution image as a vector $\mathbf{y_k}$ with length $M = M_1 \times M_2$ and the high-resolution image $N_1 \times N_2$ image as a vector $\mathbf{z}$ with length $N = N_1 \times N_2$. Low-resolution images are modeled as a linear transformation of the high-resolution image followed by a downsampling step with factor $L = M_1/N_1 = M_2/N_2$. An additive Gaussian noise random variable $\eta \sim N\left(0, \sigma_\eta^2\right)$ accounts for camera noise effects [1] The linear transformation is:

$$\mathbf{y_k} = \mathbf{W^k z} + \eta$$

Here $\mathbf{W^k}$ is an $M \times N$ matrix containing the weighted contribution of each pixel in $\mathbf{z}$ to the pixels of $\mathbf{y_k}$. $\mathbf{W^k}$ is determined by the relative movement of the low-resolution image to the reference image, specified by a translation parameter vector $\mathbf{s_k}$ and rotation matrix $\mathbf{R_\theta^k}$, and a point spread function that models the diffusion of light to pixels. To determine $\mathbf{W^k}$, we calculate each weight W_{ij}^k as the value of a Gaussian point spread function between the reference image and the shifted low-resolution image [2]:

$$W_{ij}^k = \exp(-\frac{||\mathbf{v_i} - \mathbf{u_j}||^2}{\gamma^2})$$

Here the width γ of the point spread function determines how much 'leakage' there is from high-resolution pixels to low-resolution pixels. γ is a constant that characterizes the camera system and should be chosen to match the type of sensor being used. $\mathbf{v_i}$ is the pixel location in the high resolution image and $\mathbf{u_j}$ is the center of the point spread function obtained by:

$$\mathbf{u_j} = \mathbf{R}_\theta(\mathbf{v_j} - \mathbf{v}) + \mathbf{v} + \mathbf{s_k}$$

Here $\mathbf{v}$ is the center of the image. Note that if there are no rotations involved in the construction of low-resolution images, this relationship becomes

$$\mathbf{u_j} = \mathbf{v_j} + \mathbf{s_k}$$

Given a set of $\mathbf{y_k}$ we wish to obtain the corresponding motion parameters $\mathbf{s}$ and the high resolution image $\mathbf{z}$. To simplify notation, we will construct $\mathbf{y}$, a column vector containing all k low-resolution vectors. Similarly, we denote $\mathbf{W}$ as all of the $\mathbf{W^k}$ stacked on top of each other, allowing us to write our model as $\mathbf{y} = \mathbf{Wz} + \eta$ with η now a pM-dimensional vector. We formulate the MAP estimate of $\mathbf{s}$ and $\mathbf{z}$.

$$\hat{\mathbf{z}}, \hat{\mathbf{s}} = arg \max_{\mathbf{z},\mathbf{s}} \Pr(\mathbf{y}|\mathbf{z}, \mathbf{s}) \Pr(\mathbf{z}, \mathbf{s})$$

Assuming $\mathbf{z}$ and $\mathbf{s}$ are independent, and minimizing the negative log-likelihood function, we obtain:

$$\hat{\mathbf{z}}, \hat{\mathbf{s}} = arg \min_{\mathbf{z},\mathbf{s}} L(\mathbf{z}, \mathbf{s})$$
$$= arg \min_{\mathbf{z},\mathbf{s}} \{-\log(\Pr(\mathbf{y}|\mathbf{z}, \mathbf{s})) - \log(\Pr(\mathbf{z})) - \log(\Pr(\mathbf{s}))\}$$

Figure 1. High resolution image 256×256 and test low-resolution image 64×64 with added noise

The prior on $\mathbf{z}$ was chosen to be a Gaussian to model the statistics of photons hitting a light detector [1].

$$\Pr(\mathbf{z}) = \frac{1}{(2\pi^{\frac{N}{2}})|C_z|^{1/2}} \exp\left\{-\frac{1}{2}\mathbf{z}^{\mathbf{T}} C_z^{-1}\mathbf{z}\right\}$$

We can think of the covariance matrix C_z as describing the similarity between pixels in the target image. We can rewrite $\mathbf{z}^{\mathbf{T}} C_z \mathbf{z}$ as a product $\frac{1}{\lambda}\sum_{i=1}^{N}\left(\sum_{j=1}^{N} d_{i,j} z_j\right)$, where d_i, j controls the shape of our similarity between pixels, and λ can control the weighting.

The prior of $\mathbf{s}$ is dependent on the motion characteristics of the camera system and can be tailored to specific applications. In the general case, we don't know how the system is moving, thus we will assume the prior to be a uniform distribution.

We therefore write our log-likelihood function as:

$$L(\mathbf{z},\mathbf{s}) = \frac{1}{2\sigma_\eta^2}(\mathbf{y}-\mathbf{W}\mathbf{z})^T(\mathbf{y}-\mathbf{W}\mathbf{z}) + \frac{1}{2\lambda}\sum_{i=1}^{N}\left(\sum_{j=1}^{N} d_{i,j} z_j\right)^2$$

3 Implementation

We created test low-resolution images $\mathbf{y_k}$ from a high-resolution image for training by Gaussian blurring a 268×268 high-resolution image, creating a shift (only translations) by choosing a 256×256 window, and then subsampling it by $L = 4$. Fig. 1 shows an example test image.

Since we are minimizing the likelihood function with respect to two sets of parameters, $\mathbf{z}$ and $\mathbf{s}$, one method would be to devise a coordinate descent-like algorithm, optimizing cyclically between $\mathbf{z}$ and $\mathbf{s}$[1]. We can use a gradient descent for minimizing the gradient of the log-likelihood with respect to $\mathbf{z}$. However because the dependence on $\mathbf{s}$ is implicit in our model, we don't have an expression for the derivative of the log-likelihood with respect to the shifts. So estimating the shifts are done by taking a block from the upsampled low resolution image and matching this block to the high resolution image by maximizing the 2-D correlation between the two blocks.

The algorithm begins by initializing $\mathbf{z}$ to Gaussian noise with mean zero and standard deviation σ, and initial shifts $\mathbf{s}$ to 0. We then calculate the $\mathbf{W}$ given these shifts. Given $\mathbf{W}$ and the initial value of $\mathbf{z}$ gradient descent is run to minimize the log-likelihood with respect $\mathbf{z}$. To minimize the log-likelihood we take the gradient of L($\mathbf{z}$,$\mathbf{s}$) with respect to $\mathbf{z}$:

$$\frac{\partial L(z,s)}{\partial \mathbf{z_k}} = \frac{1}{\sigma^2}\sum_{m=1}^{pM} w_{m,k}(\mathbf{s})\left(\sum_{r=1}^{N} w_{m,r}(\mathbf{s}) z_r - y_m\right) + \frac{1}{2\lambda}\sum_{i=1}^{N} d_{i,k}\left(\sum_{j=1}^{N} d_{i,j} z_j\right)$$

The first term in the gradient expression is the sum of differences between the predicted and the actual low-resolution image vectors. Each term in the sum is weighted by the contribution of $\mathbf{z_k}$ to that low-resolution pixel,$w_{m,k}(\mathbf{s})$. The second term is the prior gradient which is in fact simply a linear combination of the pixels

in the high resolution image and can also be computed via a convolution operation. Using the gradient expression given above, the update rule that this routine uses becomes;

$$\hat{\mathbf{z}}_{\mathbf{k}}^{\mathbf{n+1}} = \hat{\mathbf{z}}_{\mathbf{k}}^{\mathbf{n}} - \alpha \nabla_z L(\mathbf{z}, \mathbf{s})|_{\mathbf{z}=\hat{\mathbf{z}}_{\mathbf{k}}^{\mathbf{n}}, \mathbf{s}=\hat{\mathbf{s}}_{\mathbf{k}}^{\mathbf{n}}}$$

where $\hat{\mathbf{z}}_{\mathbf{k}}^{\mathbf{n}}$ and $\hat{\mathbf{s}}_{\mathbf{k}}^{\mathbf{n}}$ are $\mathbf{z}$ and $\mathbf{s}$ estimates at the n^{th} step and α is the step size determined by annealing, starting from relatively high step sizes and decreasing the step size as the algorithm approaches convergence.

A single gradient descent run is assumed to converge when the maximum among the absolute value of the entries in the gradient is below a defined tolerance value. After the gradient descent to the current tolerance value, the shifts are estimated by maximizing the 2-D cross-correlation of two blocks extracted from the high resolution estimate and upsampled low resolution input. The shifts giving the maximum correlation values are stores as $\hat{\mathbf{s}}_{\mathbf{k}}^{\mathbf{n}}$.

Since the accuracy of the shifts is relatively low at the initial iterations of the main loop, this tolerance value is decreased by half at every run of gradient descent starting from a predefined initial value. Thus as $\hat{\mathbf{z}}$ approaches the original value and the accuracy of the shifts increases the tolerance value is decreased, forcing the gradient descent to achieve a more accurate $\mathbf{z}$ estimate.

3.1 Rotations

We tried to incorporate rotations into the same algorithm above. Again, the gradient with respect to shift/rotation could not be calculated directly so a search through multiple cross-correlations of rotated images was performed to estimate $\mathbf{s}$ and $\mathbf{R}_\theta$. Also, updating $\mathbf{z}$ in the gradient descent step was done by rotating taking the difference between the low-res images and a rotated $\mathbf{z}$ and then rotating this difference back before applying $\mathbf{W}$. Unfortunately, the number of cross-correlations and rotations proved to be formidable in MATLAB and made the code very slow.

4 Results

The algorithm was tested using synthesized low-resolution images generated by first convolving with a Gaussian kernel and then downsampling with a predefined factor. The images generated were used as input vectors to the algorithm and the performance of the algorithm was tested for different parameter ranges. During these procedures convergence parameters of the algorithm (initial α, the initial tolerance for gradient descent convergence,)were kept constant between runs.

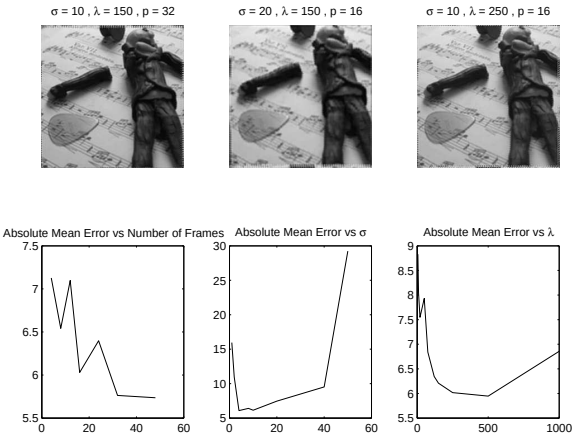

Figure 2. Example of change p, σ, and λ and the mean pixel error between $\mathbf{z}$ and original high-resolution image.

The parameters for which the performance of the algorithm was tested were σ^2 the variance of the additive gaussian noise in the low-resolution images,λ the variance of the prior for $\mathbf{z}$, p the number of low resolution images used. The performance criterion used for comparison was the absolute mean error between the resultant high-resolution estimate $\mathbf{z}$ and the original high-resolution image. Results for selected parameter sets can be seen in Figures 2 and 3.

According to the results seen in the figure the number of-low resolution images used in general increases the accuracy of the final estimate $\hat{\mathbf{z}}$. Moreover, note that the changes in σ and λ effects the magnitude of the gradient of z in the gradient descent update rule directly. Thus the effect of changing one of these parameters in general depends on the value of the other parameter. The ratio between σ^2 and λ signifies the importance of the error values over the information coming from the prior of $\mathbf{z}$ (which explains how values of nearby pixels behave with respect to each other). Thus making λ smaller with respect to σ^2 means relying mostly on the prior information and not depending on the error between the

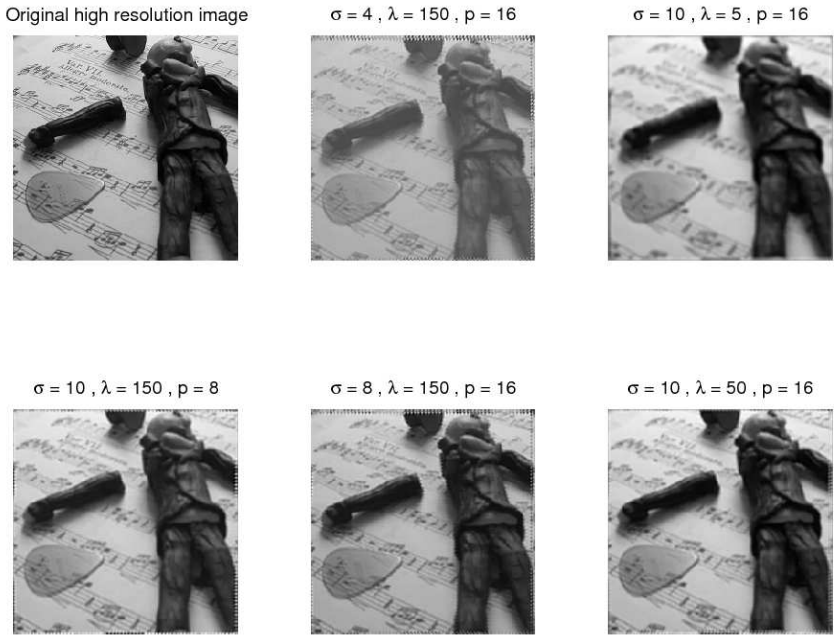

Figure 3. More estimated $\mathbf{z}$'s varying p, σ, and λ.

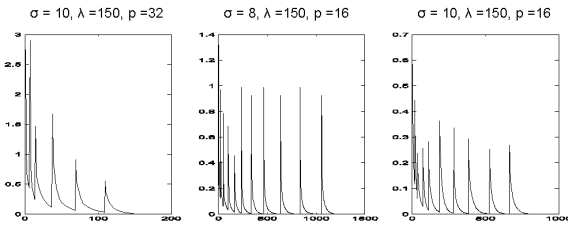

Figure 4. Learning curves for different parameter values

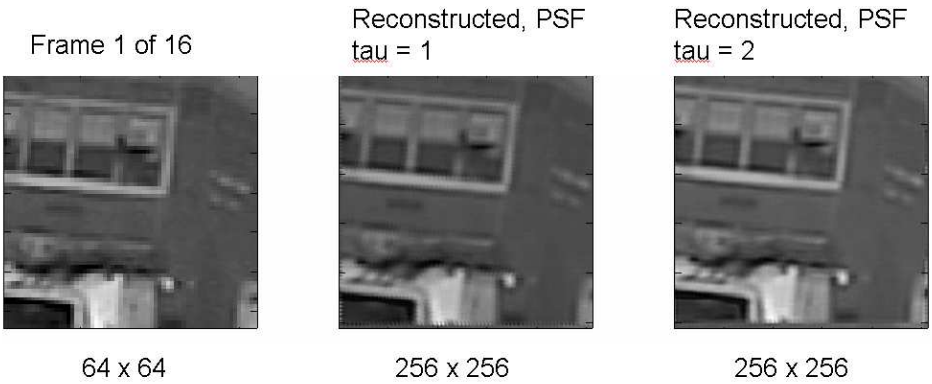

Figure 5. Training on real video shot by handheld digital video camera.

estimated and original low-resolution images.

The learning curves for some parameter combinations can be seen in Figure 4. The spiking behaviour in the curves is due to the tolerance criteria used. Since the shifts are estimated again before each gradient descent run, this changes the W matrix used during the gradient descent, making the initial delta much bigger than the previously obtained lowest value. At each run of gradient descent, the accuracy of the shifts increases. When the shift values are close to convergence the spikes disappear allowing the gradient descent to continue from the previously obtained smallest maximum absolute value.

The algorithm was also tested on some videos by taking 64×64 blocks from each frame as the low resolution input images. For these inputs because we dont have any prior information on the point spread function used, effects of changing τ value (the width of the point spread function) was observed. The results can be seen in Fig. 5. As can also be observed from the figure changing τ didnt improve the result to a great extent although some minor artifacts (vertical lines) seen for $\tau = 1$ are canceled for $\tau = 2$.

We speculate that the poor results from video footage can be explained by our failure to capture rotations in the low-resolution images. As mentioned in the previous section, attempting to incorporate rotations into our algorithm led to inconclusive results because of the increased run-time and inaccuries in estimating the rotation between frames. In Fig. 6, we show the results of attempting to estimate $\mathbf{z}$ from shifted and rotated test images. Inaccuracies in our rotation estimator made it very difficult to run the algorithm to convergence, causing the resulting estimate to be much worse than our estimates from a shifted-only training set.

Figure 6. Estimate $\mathbf{z}$ from rotated test images.

5 Conclusion

We found that estimating $\mathbf{z}$ with a MAP framework proved to be very successful when we generated test images our selves. However, with real video images, we could not successfully capture all motion parameters into the model and thus found the $\mathbf{z}$ estimates to be at best, smoothed interpolated versions of the low-resolution images. If a more accurate, computationally efficient estimate of rotation could be found, we would think the MAP estimator would be a good solution the super-resolution problem.

6 References

[1]R.C. Hardie, K.J. Barnard, E.A. Armstrong. Joint MAP registration and high-resolution image estimation using a sequence of undersampled images. IEEE Transactions on Image Processing, 6(12):1621-1633, 1997.

[2]M.Tipping,C.Bishop. Bayesian Image Super-resolution. Advances in neural information processing systems 15:1279-1286. MIT Press,Cambridge,MA,2003.