Sparse Solvers for Poisson Seamless Cloning

Ben Humberston[*]
University of British Columbia

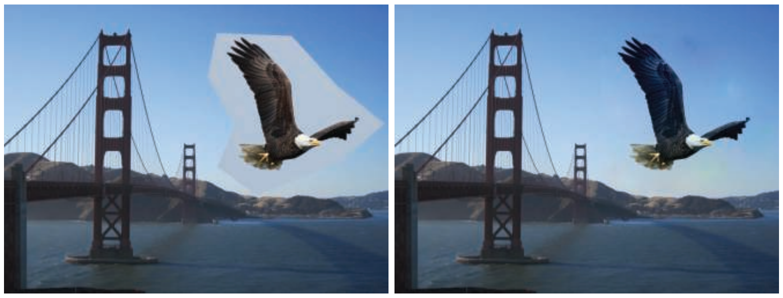


Figure 1: *Comparison of the results of basic pixel cloning (left) with seamless cloning using Poisson image editing (right)*

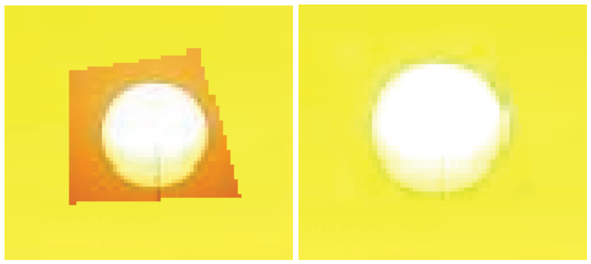


Figure 2: *Enlarged image showing domain boundary detail for basic (left) and seamless cloning (right) for a source image of the sun imposed onto the yellow sky of a destination image. The basic cloning suffers from a sharp gradient change at the border of the cloned region, visible as a four-sided polygon in this image. Seamless cloning hides this gradient discontinuity by merging the source gradients rather than its pixel values directly.*

Abstract

In image processing applications, a common problem entails merging a region from a source image into a corresponding portion of a destination image without producing visual artifacts at the edges of the cloned region. Poisson image editing is one recent technique that accomplishes this seamless cloning. It uses a Poisson formulation that equates the Laplacian of the source and destination images in the region of interest in order to smoothly transition from the destination image pixel values at the border to the source pixel values on the interior. The interior pixel values must satisfy a sparse system of equations with nonhomogeneous Dirichlet boundary conditions. This work describes the implementation of and explores the relative performance for several different iterative sparse solvers as applied to an instance of the Poisson seamless cloning problem at various scales. The solvers for comparison include successive over-relaxation, preconditioned conjugate gradient, and a multigrid V-cycle solver.

Keywords: sparse matrices, image processing, successive over-relaxation, preconditioned conjugate gradient, multigrid

1 Introduction

The cloning problem in image processing poses the question of how one may merge some portion of a natural image into another without visible seams. By “seams”, we refer to the perceptual discontinuity at the border of a cloned region where a human observer may clearly delineate between the source and destination pixels. Two-dimensional images of natural scenes (eg: landscapes, family portraits) usually have no simple analytic formula to describe the intensity value of each pixel due to the high frequency detail of the image. High frequency content arises from detailed material textures, lighting within the scene, and noise in the image acquisition process itself, among other factors. As a result, basic cloning, which directly replaces destination pixel values in the cloned region with those from a source image, will create visible seams due to the sharp change in pixel intensity at the border of the region. See the left side of Figure 2 for an example of this type of seam.

The Poisson image editing tools proposed by [Pérez et al. 2003] are designed to eliminate this issue. Their work introduces the notion of a *guidance vector field*, which dictates the new pixel gradient values of the destination image within the cloned region. Rather than importing pixel values directly from a source, Poisson image editing instead sets pixel values by painting inward from the existing border pixels of the destination image using the guidance vector field to shape the gradients. Although the Poisson image editing framework yields numerous image processing tools, we shall focus on the seamless cloning tool in particular. For this operator, the guidance vector is defined by the *source image gradient field*. Thus, Poisson seamless cloning determines the clone region pixel intensities by fixing the pixel values at the region border and setting interior pixels such that the source image gradient is preserved. Another example of results obtained using the seamless cloning tool as opposed to basic cloning is shown at the top of this page in Figure 1.

In order to solve for the new pixel intensities in the clone region, a sparse linear system is constructed, the details of which are discussed below in Section 2. This sytem may be solved using iterative methods for sparse matrices. Three well-known solvers in particular are implemented and presented in this work. The basic stationary method of successive over-relaxation (SOR) as described in [Ascher and Greif 2011] is given as a baseline for the other solvers. Because the Poisson seamless cloning problem may be formulated using the 2D symmetric positive-definite discrete Laplacian matrix,

[*]e-mail: bhumbers@cs.ubc.ca

the conjugate gradient with optional preconditioner (PCG) is applicable to the problem as well. Finally, because it is natural to consider the cloning problem at various image scales, a multigrid V-cycle solver (MG) is presented that correctly accounts for both border and interior pixel constraints. Further information on the implementation of these solvers is given in Section 3.

Quantitative comparisons of the performance of these solvers on an instance of seamless cloning are provided in Section 4. Particular interest is given to the relative performance measures when the different solvers are applied to problems of various scales (ie: larger or smaller regions for which new pixel values must be determined). Unsurprisingly, the multigrid method, which solves systems in a number of iterations independent of the problem size, proves much more viable than SOR and unpreconditioned CG for very large cloning regions. A brief discussion of the implications of the experimental results and concluding remarks are presented in Section 5.

1.1 Related Work

Discrete solution methods to the Poisson equation are frequently recruited for image processing problems, which generally involve computations on a regular two-dimensional pixel grid with well-defined gradients. [Pérez et al. 2003] notes several applications of the Poisson equations to image processing, including HDR image compression ([Fattal et al. 2002]) and object suppression or addition ([Elder and Goldberg 2001]).

An implementation of Poisson seamless cloning is provided by [Hu]. However, the problem setup is unoptimized, and the system solve is effected using the default MATLAB backslash operator. Additionally, no qualitative or quantitative performance results are given. The current work, in contrast, focuses on implementing and deriving quantitative performance results for several sparse solvers when applied to the seamless cloning problem. Finally, the implementation in [Hu] uses the "alternative", non-symmetric formulation of the coefficient matrix A (see equation 6) and so is incompatible with the preconditioned conjugate gradient method (this was verified in practice; see Section 3).

2 Poisson seamless cloning formulation

Here, specific details are given on how the seamless cloning problem is expressed and solved as a sparse linear system base on the Poisson equation.

Note that the following discussion assumes scalar pixel values. If an image has multiple color channels (eg: 3 channels for an RGB image), each channel is determined separately; the final merged image results from displaying together the independently solved color channels. In this project's implementation, all channels are solved in parallel using multiple right-hand sides and solution vectors. Although there are potential performance gains from employing techniques designed for problems with the same coefficient matrix and multiple right-hand sides, this project limits itself to solvers that only operate on one channel at a time. In fact, all performance results in Section 4 use a single-channel image for simplicity.

The Poisson seamless cloning problem requires that we determine new pixel values inside a clone destination region that follow the gradient values from some source image g, while fixing the existing destination border intensities. [Pérez et al. 2003] outlines the minimization problem associated with using a guidance vector field v to interpolate the values from a border $\partial\Omega$ into a region Ω in the continuous domain. Integrated across all points in the clone region Ω, we wish to set new intensity values f that minimize the difference between the intensity gradient ∇f and the corresponding value of the guidance field v, subject to the constraint that the intensity values f on the border of the clone region remain unchanged from their current destination image values, f^*.

$$\min_f \iint_\Omega |\nabla f - v|^2, \text{with} f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{1}$$

From this, we see that the titular "Poisson" for this technique arises from the fact that the solution to equation 1 is also the unique solution to the Poisson equation with inhomogeneous Dirichlet boundary conditions:

$$\Delta f = \text{div} v \text{ over } \Omega, \text{with} f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2}$$

In the case of seamless editing, we simply set the guidance field v to be the gradient of the source image g:

$$v = \nabla g \tag{3}$$

Thus, we have specified a minimization problem whose solution will satisfy our twin constraints that first, the border pixels must remain unchanged from their current value f^*, and second, the interior pixel gradients must equal the gradient from the source image g.

The discrete solution to the preceding continuous problem, adapted from [Pérez et al. 2003], satisfies the following system of linear equations:

$$\text{for all } p \in \Omega : 4f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f_q^* + \sum_{q \in N_p} (g_p - g_q) \tag{4}$$

Here, N_p is the set of 4 neighboring pixels of p, f_p^* indicates the values of the destination image at p, g_p is the corresponding value of the source image at p, and f_p is one of the pixels in the cloned region Ω whose intensity value we must set. Gradients are approximated at each pixel using finite differences, which gives us the expression $g_p - g_q$ on the right-hand side to approximate ∇g at the point $\frac{p+q}{2}$. The first sumation term on the right-hand side enforces the Dirichlet boundary conditions for pixels adjacent to the border of the clone region, $\partial\Omega$. This system of equations may be expressed in matrix notation as:

$$Af = b \tag{5}$$

A is the standard 2D discrete Laplacian matrix of size N x N, where N is the number of pixels in the clone region. The right-hand side b is simply the finite approximation of the source gradient ∇g for all non-border pixels; border pixels have the fixed destination value of their neighbors outside the clone region added on top of this gradient value, as shown in equation 4.

2.1 Alternative Formulation

Equivalent in its effect to the above setup, we may create an alternative system of equations in which we *include* the fixed boundaries in the linear system solution, rather than encoding them in the right-hand side vector [Jeschke et al. 2009]. To do so, we modify the standard Laplacian matrix A so that rows corresponding to border

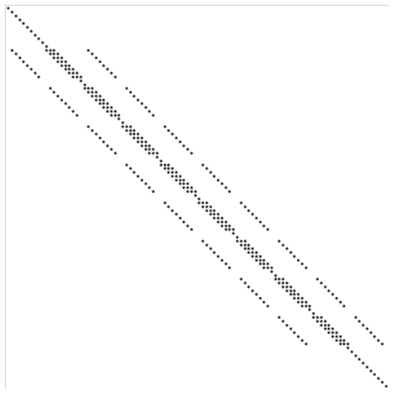


Figure 3: *The sparsity structure for a 10x10 example of the alternative coefficient matrix for a Poisson seamless cloning problem. Note that rows corresponding to border elements (the first and last 10 rows as well as every 10th interior row) have only a diagonal element, which is set to 1 (identity). This matrix is positive definite, but not symmetric*

pixels become simple identity rows with 1 on the diagonal and 0 in all other columns. The system of equations is then:

$$\text{for all } p \in \Omega \setminus \partial\Omega : 4f_p - \sum_{q \in N_p} f_q = \sum_{q \in N_p} (g_p - g_q) \qquad (6)$$

$$\text{for all } p \in \partial\Omega : f_p = f_q^*$$

This format has the virtue of simplifying the right-hand side by separating the fixed boundary constraints from the interior gradient constraints. It is the format used by [Hu], with acceptable results. However, because the matrix for this system is non-symmetric (see Figure 3), we do not expect the conjugate gradient method to converge when applied to instances of the problem in this format. In practice, it was found that applying PCG to a test image problem at various scales yielded solution estimates that tended to either diverge or approach a solution whose residual norm was well above tolerance. As such, the PCG implementation in this project uses the standard Laplacian matrix formulation from [Pérez et al. 2003], though the alternative system showed better performance when using the multigrid solver.

3 Implementation

In this section, we outline details related to the setup for a seamless cloning problem and to the implementation of each iterative solver.

3.1 Domain Shape Regularization

In the specification of the seamless cloning problem given by [Pérez et al. 2003], it is not immediately clear how to convert the irregularly shaped 2D region Ω into a column vector f whose elements correspond to the entries of a normal 2D discretized Laplacian matrix. The region is defined by a user-supplied binary mask image, which in general has arbitrary shape and a variable number of pixels in each row or column of the image grid; consequently, a typical reshaping operation that stacks each column of the domain (assumed to all be of constant size) in one vector is inapplicable without modification to the domain.

Although there are methods for solving the Poisson equation with inhomogeneous internal boundary conditions that produce an irregular domain (for example, see [Grady et al. 2005]), we may instead use the particular properties of the seamless cloning problem to modify the region of interest Ω so that the new domain is formed in a rectangular shape. As shown in Figure 4, we redefine the border $\partial\Omega$ to be a bounding rectangle that includes all pixels of the original domain.

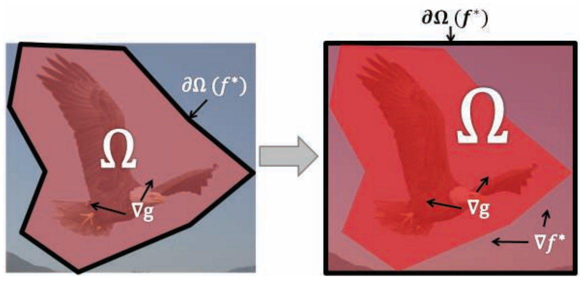


Figure 4: *A diagram demonstrating how the clone region Ω is expanded from an irregularly shaped mask region into a rectangular grid. Inside the source mask (which determines the original Ω), the guidance vector field is still defined by ∇g, the gradient of the source image. When the border $\partial\Omega$ is expanded to encompass the whole rectangular region, all newly included pixels between the old and new borders are set to use ∇f^*, the gradient of the destination image, as their guidance field.*

Expanding the border in this fashion will require that we add all of the pixels between the old and new boundaries to the new domain Ω. Originally, for pixels strictly interior to the old domain, the right-hand side vector b was set to the gradient of the source image, ∇g (see equations 4 and 6). However, we may not assign ∇g to new right-hand side entries, as the source image gradient outside the mask may contain image features that the user does not wish to include in the cloned image. Instead, we set the right-hand side for these newly included pixels to be ∇f^*, the original gradient values of the destination image. Since gradient fields are conservative and a unique solution thus exists for each seamless cloning problem, it is clear that the value that any solver assigns to these pixels will remain unchanged from their current values on the destination image, f^* (ie: starting from the fixed border values and painting inward using ∇f^*, the solution for each of the new interior pixels p will be just its original value f_p^*).

This regularization of the clone region shape adds a number of pixels to the linear system domain, increasing the size of our problem. If we so desire, we may "warm-start" our initial solution guess f_0 by setting its values to the corresponding f^* intensities for all the newly added pixels, which may reduce the initial residual norm. However, as this project's primary intent is to explore the performance of iterative sparse solvers rather than to opitimize the seamless cloning operator itself, we do not implement this or other related optimizations.

3.2 Optimal SOR relaxation parameter

When using the SOR method to solve a problem of the form $Af = b$, we must choose a value for a relaxation parameter ω somewhere in the range (0,2) to guarantee convergence of the scheme [Kahan 1958]. If $\omega = 1$, SOR reduces to Gauss-Seidel iterations. If $\omega < 1$, the iterations are damped, which may help the convergence of currently diverging iterations. However, if our iterations are already converging, we may choose $1 < \omega < 2$ in order to accelerate the convergence rate. Assuming that we use the standard discrete Laplacian version of the coefficient matrix A for the seamless cloning problem, we can derive explicitly the optimal parameter choice ω_{opt}, as outlined in [Yang and Gobbert 2009]:

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho_J^2}} \tag{7}$$

Here, ρ_J is the spectral radius of the Jacobi iteration matrix $T = I - D^{-1}A$ (D being the matrix consisting only of the diagonal elements of A). For the discrete 2D Laplacian matrix A, $\rho_J = \cos\frac{\pi}{n+1}$, where n is the number of pixels in one dimension of the grid, which varies with the size of the clone region. The number of iterations until convergence is expected to be minimized when using ω_{opt}. In Section 4.1, we verify this fact experimentally.

3.3 PCG Preconditioners

While the conjugate gradient method alone offers passable performance, it is typical in practice to apply a preconditioner matrix M to the system $Af = b$, yielding the new system:

$$M^{-1}Af = M^{-1}b \tag{8}$$

To minimize the cost of applying M^{-1} to Af, M is often chosen to be a triangular or otherwise easily invertible matrix. While the preconditioner solve adds a computational cost to each CG iteration, it is included in the algorithm with the intent of reducing the total number of iterations by a more-than-compensating factor. This iteration shaving is possible when the matrix $M^{-1}A$ has a lower condition number than A alone, or when its eigenvalues are more closely clustered than those of A [Ascher and Greif 2011].

For the purposes of this project, we test Jacobi, Gauss-Seidel, and level 0 and threshold versions of the incomplete Cholesky decomposition (abbreviated as IC(0) and ICT(0.001), respectively). The Jacobi preconditioner, similar to the splitting matrix M for stationary Jacobi iterations, is simply the diagonal matrix D constructed from the main diagonal of A. The symmetric variation of the Gauss-Seidel preconditioner matrix is split into matrices M_1 and M_2 to avoid the cost of matrix multiplication [Saad 2003]:

$$\begin{aligned} M_1 &= L \\ M_2 &= D^{-1}L^T, \end{aligned} \tag{9}$$

where L is the lower triangular portion of A, including the diagonal. To use a split preconditioner of this form, M_1^{-1} is applied first, followed by M_2^{-1}.

The IC(0) preconditioner is a Cholesky decomposition FF^T of A, except that the sparsity pattern of A is imposed on the resulting matrix F. Similarly, the ICT(0.001) preconditioner is a Cholesky decomposition in which elements of F below the threshold of 0.001 are dropped to 0. The drop threshold was chosen based on informal experimentation with the ICT preconditioner that found the best PCG performance at this value.

Quantitative performance results for the various preconditioners for the PCG method are given in Section 4.2.

3.4 Multigrid

Multigrid solvers are designed to take advantage of the ability to recast some problem types at multiple resolutions. They run a small number of smoothing iterations at each level of resolution and recursively compute a correction factor to the current residual r by descending to a coarser resolution discrete grid (see Chapter 13 of [Saad 2003] or [Agrawal and Raskar 2007] for algorithm details). When converting the residual from the fine to coarser level, a *restriction* operator reduces the residual vector resolution; conversely, when returning a correction vector from a coarse level upward to a finer one, a *prolongation* operator is applied. This project uses V-cycle multigrid, a version in which the correction term is recursively computed a single time in between smoothing iterations. Damped Jacobi is employed for the smoothing iterations.

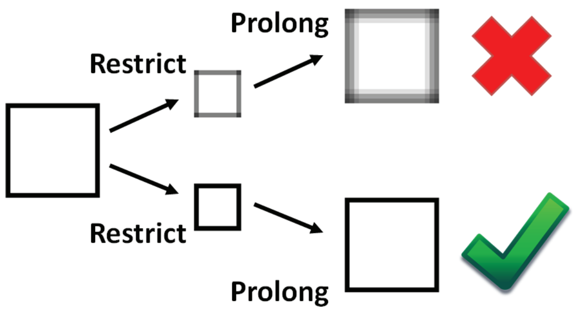


Figure 5: *When using the alternative Laplacian formulation with the multigrid solver, it is important that the restriction and prolongation operators do not mix the gradient pixels (white interior) with the boundary intensity values that enforce the Dirichlet boundary conditions (black border). The top path shows the undesired mixing that occurs when using normal bilinear filtering. The bottom path demonstrates correct separation of borders and the interior via separate filtering passes.*

A particular issue that emerged during implementation of the multigrid method was that the alternative formulation of the seamless cloning problem given by equation 6 does not converge when using standard bilinear interpolation to restrict the residual or prolong the correction vectors. The reason for this is illustrated in Figure 5. When the restriction and prolongation operators use bilinear filtering across the whole residual/correction image, the border pixel values "bleed" inward due to the 3x3 kernel of the filter (similarly, the interior gradient pixels diffuse outward to contaminate the border pixels). If border pixels are gradient values like the interior pixels, this bleeding does not pose an issue. However, because the elements on the domain border $\partial\Omega$ are fixed to destination pixel intensities f^* rather than pixel gradients, they must be preserved (ie: interpolated) separately from the interior gradients when restricting the residual. A small modification of the prolongation operator that interpolates the interior points and borders in separate passes is sufficient for this purpose.

Another difficulty that may appear during restriction is loss of magnitude when computing the coarsened residual vector. During the latter iterations of multigrid, the residual tends to have a sparse pattern that is zero in most elements but significantly far from zero in the remaining elements. If a standard intensity-preserving bilinear resampler is used to restrict the residual (using the analogue of the 1D filter [1/4, 1/2, 1/4]), the remaining non-zero elements are mixed heavily with neighboring zero elements, which causes residual pixel values to trend closer to 0 at each level of recursion. As a result, the corrections computed at the coarsest levels of the grid are underestimates of the true correction needed, and the multigrid solver becomes reliant on the damped Jacobi iterations to find a solution, which may require hundreds of V-cycle iterations, depending on the problem size (this was an observed dysfunction during the implementation of this solver).

To prevent this misbehavior, the restriction operator injects magnitude into the coarsened residual by multiplying its interpolated values by four (as recommended by [McCann and Pollard 2008]). After this correction is made, we are able to observe happily the hallmark feature of multigrid, that is, a near-constant number of V-cycle iterations that is independent of the problem size.

The multigrid method requires that we choose parameters ν_1 and ν_2, which specify the number of damped Jacobi iterations to run before and after the recursive V-cycle call, respectively. In this instance, they were chosen empirically by observing the multigrid solver's performance with different combinations of the parameters. Surprisingly strong performance was achieved when ν_1, the number of pre-smoothing iterations, was set to 0, though further research showed this is not entirely unusual for gradient image processing applications [McCann and Pollard 2008]. The values which gave the best performance, though, were $\nu_1 = \nu_2 = 100$; fewer iterations caused V-cycles to be faster but ineffective (and led to residual divergence when $\nu_2 < 10$) and more iterations increased the cost of V-cycles too much to be offset by the increased smoothing.

4 Results

The following provides quantitative results for the application of SOR, regular and preconditioned conjugate gradient, and the multigrid method to an instance of the seamless cloning problem. A 2500 x 1700 source image of an eagle in flight (provided by [Birdwatchingdaily.com]) is superimposed onto a portion of the sky of a 4000 x 3000 image of the Golden Gate Bridge (personal collection). The source image was chosen for its detailed subject on a relatively flat background. The destination image was selected for its broad sky section over which there is a vertical light-to-dark blue color gradient, which is handled particularly well by seamless cloning. While this image pair is perhaps not the most visually striking application of seamless cloning, it is advantageous for performance testing, as the correctness of a cloning result is quickly verified by visual inspection (see Figure 10 at the end of this document).

The area of the source image selected for seamless cloning is 1685 x 1843 at full resolution, which results in roughly 3.1 million pixels for which we must solve. Note, however, that we pre-scale down the source, mask, and destination images using bicubic filtering to accelerate the runtime for most tests. For example, scaling the image dimensions by 1/4x results in a problem approximately one-sixteenth the size of the unscaled problem (reducing the number of pixels to solve to 194,000).

To simplify the performance measures, we convert the 3-channel RGB problem into a monochrome problem by limiting our cloning operation to only the red (R) channel of the source and destination image. Thus, there is a scalar stopping criteria and residual norm at each iteration as opposed to a 3-element vector. Performance when using a 3-channel image is similar to the 1-channel results, excepting of course that the total solve takes roughly three times as long as there are triple the channels that must be independently determined (the number of iterations before convergence tends to be roughly the same for each channel).

We begin our performance evaluation by first determining or confirming the optimal parameters for each category of solver. Following these preliminary evaluations, we compare the performance of solvers across different categories.

4.1 SOR Relaxation Parameter

We emprically verify our choice of the optimal relaxation parameter ω_{opt} by observing its convergence behavior compared to a set of other nearby values of ω. It should be noted that convergence was

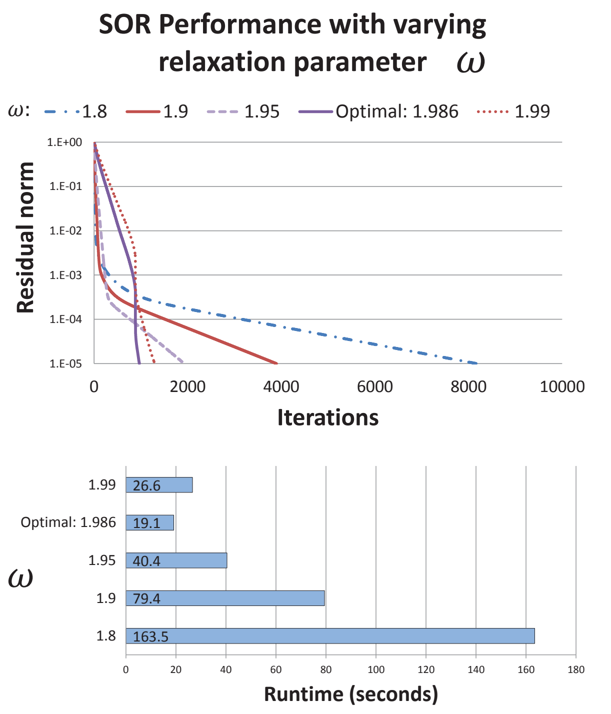


Figure 6: *Residual norm per iteration (top) and runtimes (bottom) for the SOR solver with various values of ω when applied to the test image problem at 1/4x resolution.*

not achieved within a 20 minute time limit when SOR was applied the the test image at 1/4x scale for any $\omega \leq 1.7$, which underscores the importance of choosing a suitable relaxation parameter. Figure 6 shows the residual norms at each iteration and total runtime for a few ω values in the vicinity of $\omega_{opt} \approx 1.986$. It is clear that SOR's convergence rate rapidly improves for values near to ω_{opt}, and we are able to confirm that the expression for ω_{opt} given by equation 7 indeed specifies the best choice of ω.

4.2 PCG Preconditioner

The relative performance of different preconditioners (or lack thereof) for the PCG method are shown in Figure 7. Regular CG (no preconditioner) and the Jacobi preconditioner show nearly identical, poor performance, while ICT(0.001) requires the fewest iterations before convergence. The runtimes reflect the same performance ordering as the residual norm trends: using either IC(0) or Gauss-Seidel is roughly 2x slower than the ICT(0.001) preconditioner, and normal CG and using a Jacobi preconditer are about 2x slower still.

4.3 Multigrid

Informal performance testing revealed that the alternative seamless cloning formulation of equation 6 ran faster on average than the standard version from [Pérez et al. 2003] for the multigrid solver. As such, the multigrid solver performance measures were produced using the alternative system; in contrast, performance measures for SOR and PCG both use the standard formulation.

The effect of using different interpolation methods for the restriction and prolongation operators was also briefly considered. When resampling a residual or correction vector, popular methods include nearest-neighbor filtering, which is very fast but is likely to

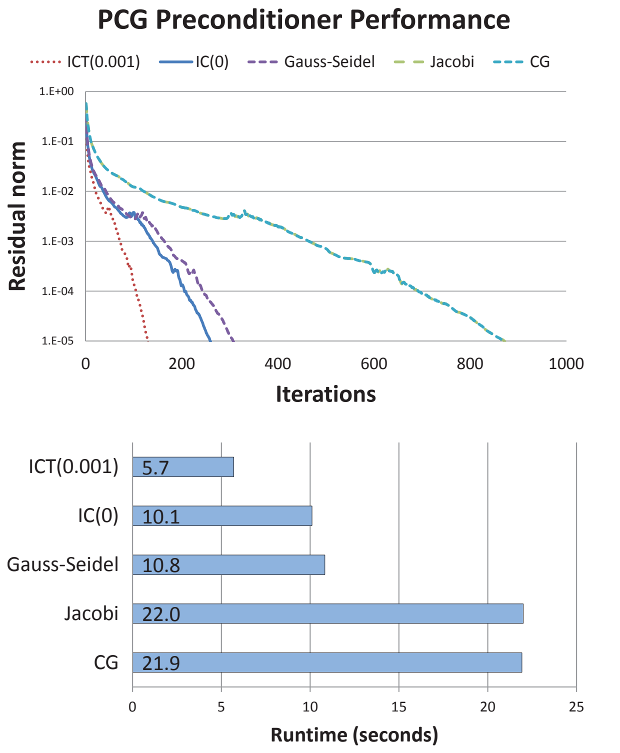


Figure 7: *Residual norm per iteration (top) and runtimes (bottom) for the PCG solver with various preconditioners when applied to the test image problem at 1/4x resolution.*

cause aliasing, bilinear interpolation, and bicubic filtering. Bicubic filtering offers increased smoothness over bilinear filtering at the cost of a wider kernel that must merge more source pixels. On the 1/4x scale test image problem, the nearest-neighbor filtering method failed to converge, unsurprisingly. The iteration count and runtime of the multigrid solver did not significantly differ between bilinear or bicubic interpolation.Thus, bilinear interpolation was chosen by convention as the interpolation method for the restriction/prolongation operators.

4.4 Comprehensive: Fixed Problem Size

Having determined the value and verified the efficacy of our chosen parameters in each individual solver category, we are prepared to compare the performance of the solvers across categories. Figure 8 shows the results when the SOR, CG, PCG with ICT(0.001) preconditioner, and multigrid solvers are applied to the test problem at 1/4x scale. Multigrid and PCG both show make strong showings, while SOR and un-preconditioned CG exhibit slow convergence for a problem of this size.

4.5 Comprehensive: Variable Problem Size

A final experiment compares the runtime trends among the solvers when applied to problems of increasing dimension. Each solver is applied to the test image problem at scales ranging from 1/16x scale to full size. The results are shown in Figure 9; we note that they are generally are comparable with those of similar experiments from [Ascher and Greif 2011], p. 209.

At 1/16x scale, the test problem contains about 12,400 pixels that must be determined; as noted above, at full scale, the clone region contains about 3.1 million pixels. Each solver shows a distinctive

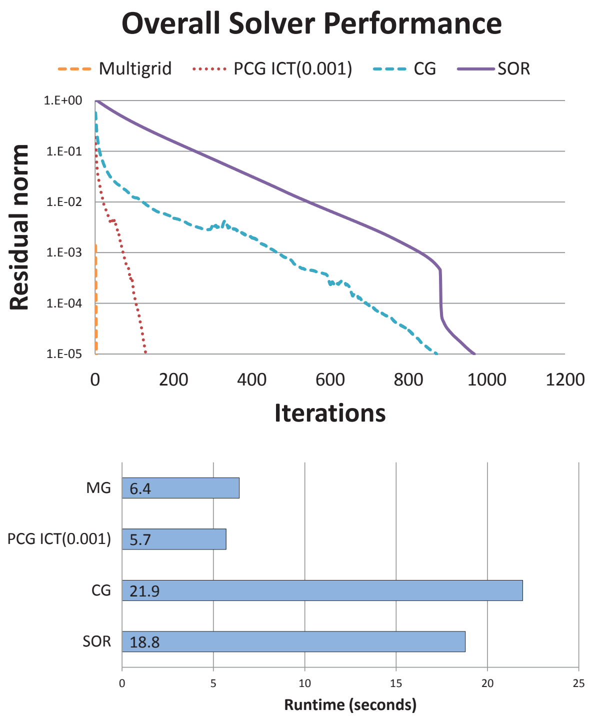


Figure 8: *Residual norm per iteration (top) and runtimes (bottom) of SOR, regular CG, PCG using the ICT(0.001) preconditioner, and multigrid when applied to the test image problem at 1/4x resolution. Multigrid clearly requires the fewest iterations, yet its runtime is slightly longer than PCG with ICT(0.001) preconditioner.*

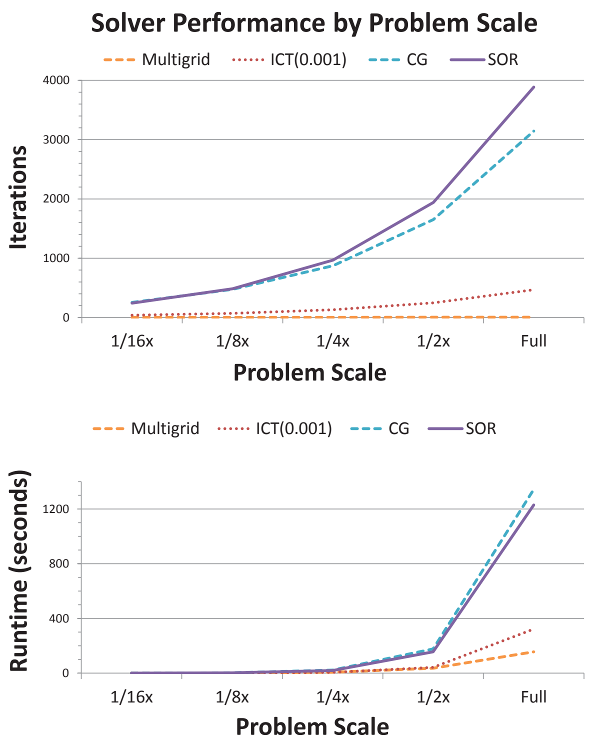


Figure 9: *Iterations required (top) and runtime (bottom) for convergence as a function of problem scale for SOR, CG, PCG using the ICT(0.001) preconditioner, and multigrad when applied to the test image problem at multiple resolutions (resolution increase from 1/16x scale to full).*

scaling behavior. At low scales such as 1/16x or 1/8x, all solvers converge within a couple seconds. However, at full scale, the runtimes of both SOR and unpreconditioned CG explode to over 20 minutes while the solve time of advanced solvers such as PCG using ICT(0.001) and multigrid runtimes remain somewhat tractable, coverging to tolerance within a few minutes.

The number of iterations required to reach convergence parallels the runtime by solver. Both SOR and unpreconditioned CG require iterations that scale approximately linearly with the problem size, in accordance with theoretical predictions. The necessary iteration count for convergence using PCG with ICT(0.001) increases with problem size as well, but at a much lower rate. The multigrid method, however, requires only a small number of V-cycle iterations regardless of problem size, as expected; the extra processing occurs within each cycle via the additional resolution levels added by the increase in problem size.

5 Discussion

Based on our above results, we may imagine that a person tasked with designing a seamless cloning image editing application would be well-advised to implement a hybrid approach that chooses which solver to use for a given cloning instance based on the size of the clone region. If the region contains only a few thousand pixels, then we have reason to prefer either SOR or unpreconditioned conjugate gradient. These solvers are simple to implement and require only a minimal increase of memory on top of the existing image data to be run, and their most complex operation is just a matrix-vector product per iteration. On the other hand, if the user requests a clone operation that covers hundreds of thousands or millions of pixels, then SOR and regular CG will not scale to the user's satisfaction. The designer of the program may consider switching to a preconditioned CG solver that uses ILU(0) or ICT(0.001) or a multigrid V-cycle method. While these solvers add complexity and additional memory requirements (space for the preconditioner matrix if explicitly formed for PCG, image pyramids of the error/correction for multigrid), they exhibit performance that scales more acceptably to large problems.

Seamless cloning between two different images is made possible by a gradient-level equality constraint between the source and destination images. As first introduced by [Pérez et al. 2003], the system of equations that determines pixel values in the clone region has a unique solution that corresponds to the solution of a Poisson problem with Dirichlet boundary conditions. The system may be solved using general sparse iterative methods. This project implemented and provided performance metrics for variations on three particular solvers: SOR, conjugate gradient with optional preconditioner, and a geometric multigrid method.

Current and future work related to gradient-level image editing includes the implementation of solvers that take advantage of massively parallel hardware, including GPUs [Jeschke et al. 2009], as well as implementations that are suitable for low-power, low-memory mobile devices [Xiong et al. 2009].

References

AGRAWAL, A., AND RASKAR, R. 2007. *Gradient Domain Manipulation Techniques in Vision and Graphics*.

ASCHER, U., AND GREIF, C. 2011. *A First Course in Numerical Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

BIRDWATCHINGDAILY.COM. Bald eagle. `http://cs.birdwatchingdaily.com/BRDCS/media/p/72070.aspx`.

ELDER, J. H., AND GOLDBERG, R. M. 2001. Image editing in the contour domain. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 3 (Mar.), 291–296.

FATTAL, R., LISCHINSKI, D., AND WERMAN, M. 2002. Gradient domain high dynamic range compression. *ACM Trans. Graph. 21*, 3 (July), 249–256.

GRADY, L., TASDIZEN, T., AND WHITAKER, R. T. 2005. A geometric multigrid approach to solving the 2d inhomogeneous laplace equation with internal dirichlet boundary conditions. In *ICIP (2)*, 642–645.

HU, R. Poisson image editing. `http://www.eecis.udel.edu/~rhu/hw2/HW2page.htm`.

JESCHKE, S., CLINE, D., AND WONKA, P. 2009. A gpu laplacian solver for diffusion curves and poisson image editing. In *ACM SIGGRAPH Asia 2009 papers*, ACM, New York, NY, USA, SIGGRAPH Asia '09, 116:1–116:8.

KAHAN, W. 1958. *Gauss-Seidel Methods of Solving Large Systems of Linear Equations*. University of Toronto.

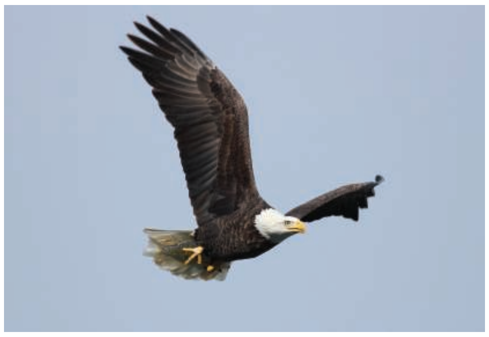
MCCANN, J., AND POLLARD, N. S. 2008. Real-time gradient-domain painting. *ACM Trans. Graph. 27*, 3 (Aug.), 93:1–93:7.

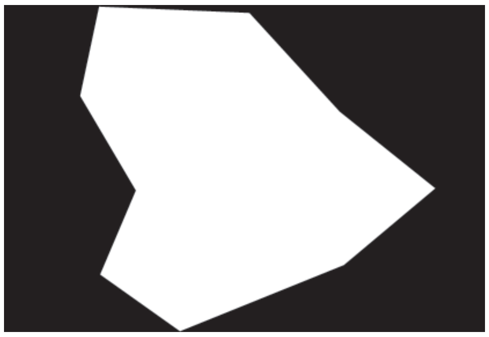
PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3 (July), 313–318.

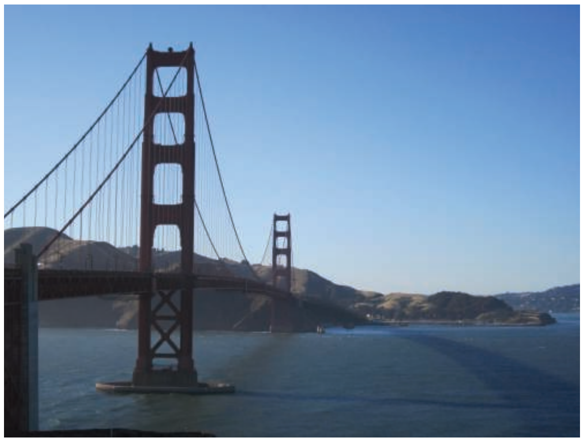
SAAD, Y. 2003. *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

XIONG, Y., WANG, X., TICO, M., LIANG, C.-K., AND PULLI, K. 2009. Panoramic imaging system for mobile devices. In *SIGGRAPH '09: Posters*, ACM, New York, NY, USA, SIGGRAPH '09, 36:1–36:1.

YANG, S., AND GOBBERT, M. K. 2009. The optimal relaxation parameter for the sor method applied to the poisson equation in any space dimensions. *Applied Mathematics Letters 22*, 325 – 331.


(a) Source Image


(b) Source Mask


(c) Destination Image

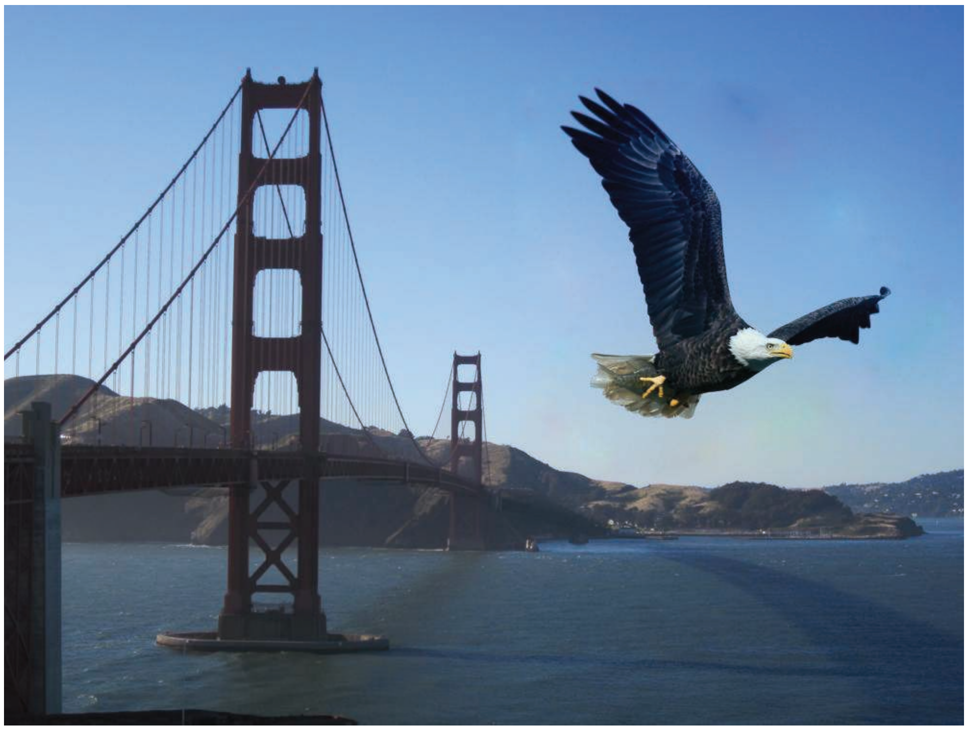

(d) Final seamlessly cloned image

Figure 10: *Images used in the performance testing of the iterative solvers. All of the solvers converge to the same final image. The "blue-ing" visible in the eagle's wings is a known side effect of Poisson seamless cloning; the merged source pixels tend to take on the color of the destination image.*