Computational Efficiency Improvements for Image Colorization

Chao Yu*, Gaurav Sharma*, Hussein Aly†*
* ECE Dept., University of Rochester, Rochester NY 14627, USA
†Military Technical College, Cairo, Egypt

ABSTRACT

We propose an efficient algorithm for colorization of greyscale images. As in prior work, colorization is posed as an optimization problem: a user specifies the color for a few scribbles drawn on the greyscale image and the color image is obtained by propagating color information from the scribbles to surrounding regions, while maximizing the local smoothness of colors. In this formulation, colorization is obtained by solving a large sparse linear system, which normally requires substantial computation and memory resources. Our algorithm improves the computational performance through three innovations over prior colorization implementations. First, the linear system is solved iteratively without explicitly constructing the sparse matrix, which significantly reduces the required memory. Second, we formulate each iteration in terms of integral images obtained by dynamic programming, reducing repetitive computation. Third, we use a coarse-to-fine framework, where a lower resolution subsampled image is first colorized and this low resolution color image is upsampled to initialize the colorization process for the fine level. The improvements we develop provide significant speed-up and memory savings compared to the conventional approach of solving the linear system directly using off-the-shelf sparse solvers, and allow us to colorize images with typical sizes encountered in realistic applications on typical commodity computing platforms.

1. INTRODUCTION

Colorization [1] refers to the problem of adding colors to monochrome images and videos. Colorizing is of considerable interest for the large collection of black-and-white movies, television shows, and photographs that were produced before color capture became widely prevalent. Also closely related is the re-colorization problem [2] in consumer photography, where image colors are transformed automatically or according to user inputs. Colorization is traditionally an extremely time-consuming task that requires a skilled artist to manually add and iteratively adjust the colors and shading in an image. Recently, promising results have been obtained with optimization based colorization algorithms that require only a sparse set of colors to be manually specified in an image from which colors are automatically propagated throughout the image while preserving natural smoothness constraints [1, 3, 4]. One example of the color scribble input and the colorization output from such an algorithm is shown in Fig. 1. Although the optimization approaches offer impressive results, efficiency improvements that reduce the computation time and memory requirements for these methods are rather desirable to allow them to be deployed for high resolution imagery and to allow them to be used interactively in situations where a user may wish to modify the colors selected for the scribbles iteratively after seeing the results of the colorization.

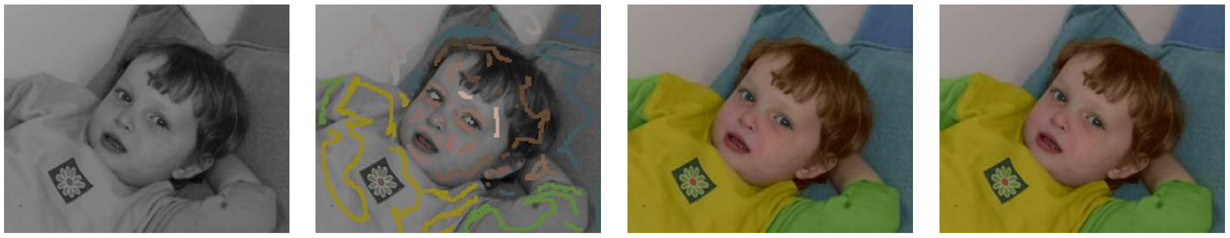
In this paper, we propose a computation and memory efficient algorithm for image colorization based on the optimization formulation for colorization proposed by Levin et. al. [1], where the optimal colorized image minimizes a quadratic cost function and can be obtained by solving a large sparse linear system of equations. Our improvements to computational efficiency derive from three innovations: (1) we consider an iterative solver for the sparse linear system based on the conjugate gradient method and avoid explicitly constructing the large sparse matrix defining the linear system, thereby significantly reducing the memory requirement, (2) we formulate each iteration in terms of integral images [5] computed via dynamic programming to reduce computational complexity by reducing repetitive computations, and (3) we accelerate the convergence of the iterative approach by using a coarse to fine approach where a lower resolution subsampled version is first colorized and upsampled to initialize the actual higher resolution colorization problem.

Computationally efficient colorization has also been previously addressed in [3] using an alternative geodesic interpolation formulation of the problem that does not use the optimization formulation that we focus on. Instead, each image pixel is colorized by interpolating the specified colors for the scribbles weighted by geodesic distances to the scribbles. While the approach achieves complexity linear in the number of pixels, the memory requirements remain high and the computational cost also increases when larger number of scribbles are required, something that is often necessary for more complex images. Also, when the image being colorized has fine-grained texture such as hair, the visual quality of the colorization degrades for the geodesic framework because this texture information is not explicitly modeled [6]. We note that the optimization formulation of colorization has also previously been accelerated in [3, 7] by using a multigrid solver

Computational Imaging XII, edited by Charles A. Bouman, Ken D. Sauer, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 9020, 902004 · © 2014 SPIE-IS&T · CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2041637

for large sparse linear system [8], however, the multigrid approach has also been observed to degrade the colorized image quality when there are few color scribbles [3, 7].

The paper follows the following organization. We present the optimization formulation of the colorization problem in Section 2 and describe our iterative method in Section 3. Results comparing the colorized images and computation and memory requirements against prior approaches are presented in Section 4 and concluding remarks in Section 5.



(a) Original grayscale image (b) Input to proposed algo. (c) Colorization (proposed) (d) Colorization of [1]

Figure 1. Colorization example. Images are best viewed electronically on a color display.

2. PROBLEM FORMULATION

We consider the colorization problem in a luminance-chrominance color space. Specifically, using standard stacked notation, the three image channels are represented as vectors $\mathbf{Y}, \mathbf{I}$ and $\mathbf{Q}$ corresponding, respectively to the luminance (Y) channel and the two chrominance channels (I & Q). The pixel values of the i^{th} pixel are then denoted by the corresponding subscripted variables Y_i, I_i and Q_i and the corresponding spatial coordinates are designated by (x_i, y_i), where $1 \le x_i \le w, 1 \le y_i \le h$, with w and h being the width and height, respectively, of the image. $N = wh$ is the total number of pixels in the image.

Given the grayscale image $\mathbf{Y}$, in addition to a sparse scribble image provided by the user, denoted by $\mathbf{I_s}$ and $\mathbf{Q_s}$, our colorization algorithm obtains estimates $(\mathbf{I}^*, \mathbf{Q}^*)$ for the true chrominance channels $(\mathbf{I}, \mathbf{Q})$. The two chrominance channels are estimated separately. We describe the estimation process for the $\mathbf{I}$ channel; the $\mathbf{Q}$ channel is treated similarly. The colorization is formulated as an optimization problem [1]:

$$\mathbf{I}^* = \arg\min_{\mathbf{I}} \lambda(\mathbf{I}^T - \mathbf{I_s}^T)\mathbf{D}(\mathbf{I} - \mathbf{I_s}) + \mathbf{I}^T\mathbf{L}\mathbf{I}, \tag{1}$$

where $\mathbf{D}$ is $N \times N$ diagonal matrix with $\mathbf{D}(i,i) = 1$ if the color scribble $\mathbf{I_s}$ is defined at pixel i, otherwise $\mathbf{D}(i,i) = 0$. λ is a large constant, and the first term on the right hand side of (1) enforces the colorization output to follow the color scribble input constraint. The second term in the right hand side enforces the local smoothness of colorization output, and $\mathbf{L}$ is an $N \times N$ matrix and referred to as the affinity matrix [1], whose construction we introduce next.

The relation between $\mathbf{I}$ and $\mathbf{Y}$ is modeled by a locally linear model

$$I_i = aY_i + b, \tag{2}$$

where a and b are constants within a neighborhood window ω_i around (x_i, y_i). We use a square window of size $(2r+1)$, so that $\omega_i \overset{\text{def}}{=} \{(x_j, y_j) | x_i - r \le x_j \le x_i + r, y_i - r \le y_j \le y_i + r\}$. The locally linear model between $\mathbf{I}$ and $\mathbf{Y}$ has the intuitive interpretation that $\mathbf{I}$ has an edge only when $\mathbf{Y}$ has an edge, since the gradients for $\mathbf{I}$ and $\mathbf{Y}$ are identical except for a scaling factor a.

The effectiveness of the local linear model (2) can be evaluated by a quadratic cost function

$$J(\mathbf{I}, \mathbf{a}, \mathbf{b}) = \sum_k \sum_{i \in \omega_k} (I_i - (a_k Y_i + b_k))^2 + \epsilon a_k^2, \tag{3}$$

where $k \in [1, N]$, a_k, b_k denotes the linear fitting parameter for the window around I_k, and the regularization parameter ϵ avoids overfitting for I_i.

The cost function (3) derived from the local linear model (2) has been studied in the image matting problem [7]. It is shown in [7] that $\mathbf{a}, \mathbf{b}$ in (3) can be eliminated by obtaining each individual optimal a_k, b_k for each window ω_k, and (3) can be reduced to

$$J(\mathbf{I}) = \arg\min_{\mathbf{a},\mathbf{b}} J(\mathbf{I}, \mathbf{a}, \mathbf{b}) = \mathbf{I}^T\mathbf{L}\mathbf{I}. \tag{4}$$

Algorithm 1 Solve the sparse linear system $\mathbf{AI} = \mathbf{b}$ using conjugate-gradient algorithm [9]

```
Input: Initial guess $\mathbf{I}_0$, convergence threshold $\tau$
Output: $\tilde{\mathbf{I}}$: estimate for $\mathbf{I}$
1: Initialize: $\tilde{\mathbf{I}} \leftarrow \mathbf{I}_0$, $\mathbf{r}_0 \leftarrow \mathbf{b} - \mathbf{A}\tilde{\mathbf{I}}_0$, $\mathbf{p}_0 \leftarrow \mathbf{r}_0$, $j \leftarrow 0$
2: while $\mathbf{r}_j^T \mathbf{r}_j > \tau|\mathbf{I}|$ do
3:     $\alpha_j \leftarrow \frac{\mathbf{r}_j^T \mathbf{r}_j}{\mathbf{p}_j^T \mathbf{A} \mathbf{p}_j}$
4:     $\tilde{\mathbf{I}} \leftarrow \tilde{\mathbf{I}} + \alpha_j \mathbf{p}_j$
5:     $\mathbf{r}_{j+1} \leftarrow \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
6:     $\beta_j \leftarrow \frac{\mathbf{r}_{j+1}^T \mathbf{r}_{j+1}}{\mathbf{r}_j^T \mathbf{r}_j}$
7:     $\mathbf{p}_{j+1} \leftarrow \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
8: end while
```

The element $\mathbf{L}(i, j)$ is obtained as

$$\sum_{k|(i,j)\in\omega_k} \left(\delta_{ij} - \frac{1}{|w_k|} \left(1 + \frac{1}{\frac{\epsilon}{|w_k|} + \sigma_k^2} (\mathbf{I}_i - \mu_k)(\mathbf{I}_j - \mu_k) \right) \right), \tag{5}$$

where δ_{ij} is the Kronecker delta function, μ_k and σ_k^2 denotes the mean and variation of pixel intensities in the window ω_k, and $|\omega_k| \overset{\text{def}}{=} (2r + 1)^2$ denotes the number of pixels in ω_k.

The solution to the quadratic optimization problem (1) is readily seen from the first order optimality condition to corresponds to the solution of the large linear system

$$(\mathbf{L} + \lambda \mathbf{D})\mathbf{I} = \lambda \mathbf{I}_s \tag{6}$$

However, the $N \times N$ matrices $\mathbf{L}$ and $\mathbf{D}$ are quite large for practical image sizes, e.g., $10^6 \times 10^6$ for a mega-pixel image. The fact that these matrices are sparse, can, however, be advantageously exploited, an approach that has been adopted in prior work on colorization where the sparsity based on direct solution of (6).

Despite the sparse structure of $\mathbf{L}$, the memory and computation requirements of direct approaches for solving (6) can be quite large, severely limiting the size of images for which the colorization can be performed on typical computers. In this paper, we develop an efficient iterative solution for (6). Compared with typical iterative methods for solving sparse linear-systems [9] we introduce three novel improvements exploiting the specific structure of our problem: the matrix $\mathbf{L}$ is not explicitly constructed or stored, integral images [5] are used with dynamic programming to eliminate repetitive computation, and a coarse to fine approach is used to provide a good initialization. Together, these improvements provide a significant improvement in both computational and memory efficiency over the prior approaches.

3. CONJUGATE GRADIENT SOLUTION USING DYNAMIC PROGRAMMING

We solve the linear systems (6) using the conjugate-gradient (CG) algorithm [9], incorporating in the process two innovations developed for the image matting problem [10], which is closely related to our colorization problem. Algorithm 1 summarizes the CG iterative procedure, where we define $\mathbf{A} = (\mathbf{L} + \lambda\mathbf{D})$ and $\mathbf{b} = \lambda\mathbf{I_s}$. The first important observation is that we do not need to explicitly construct the matrix $\mathbf{A}$, since only $\mathbf{AI}_0$ and $\mathbf{Ap}_j$ need to be evaluated at each iteration, where $\mathbf{I}_0$ is the initial guess and $\mathbf{p}_j$ is the conjugate direction in the j^{th} iteration. Our algorithm obtains $\mathbf{AI}_0$ and $\mathbf{Ap}_j$ directly, without constructing $\mathbf{A}$ explicitly, *therefore only requires a fraction of the memory required by alternative methods.*

The i^{th} element $(\mathbf{Lp})_i$ of the vector $\mathbf{Lp}$ can be obtained as [10]:

$$a_k^* = \frac{1}{\frac{\epsilon}{|w_k|} + \sigma_k^2} \left(\frac{1}{|\omega_k|} \sum_{i\in\omega_k} I_i p_i - \mu_k \bar{p}_k \right), \tag{7}$$

$$b_k^* = \bar{p}_k - a_k^* \mu_k, \tag{8}$$

$$(\mathbf{L}p)_i = |\omega_i| p_i - \left(\sum_{k\in\omega_i} a_k^*\right) I_i - \left(\sum_{k\in\omega_i} b_k^*\right), \tag{9}$$

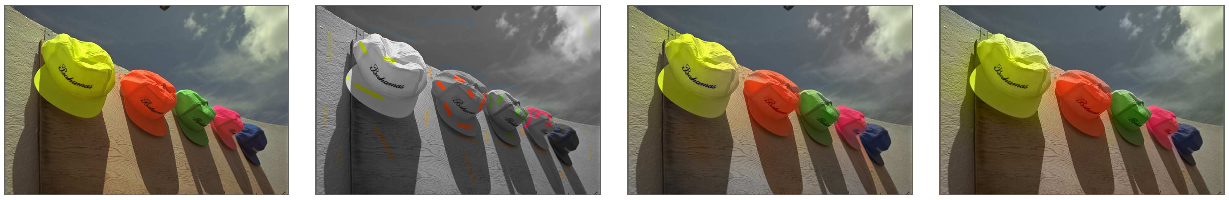

(a) Original color image (b) Input to the proposed algo. (c) Colorization (proposed) (d) Colorization [1]

Figure 2. Additional colorization result.

where μ_k and $\bar{p}_k$ denotes the average value of $\mathbf{I}$ and $\mathbf{p}$ in the window ω_k respectively. Equations (7)- (9) correspond to a scalar version of the analysis in [10], since the $\mathbf{I}$ channel being estimated is a greyscale signal. For the matrix $\mathbf{A} \overset{\text{def}}{=} (\mathbf{L} + \lambda\mathbf{D})$, the vector $\mathbf{Ap}$ can then be obtained by $(\mathbf{Ap})_i = (\mathbf{Lp})_i + \lambda\mathbf{D}(i,i)\mathbf{p}_i$.

Equations (7)- (9) suggest that in order to calculate $\mathbf{Ap}$, for each local window, we need to calculate the summations for several quantities, $\sum_{k\in\omega_i} a_k^*$, $\sum_{k\in\omega_i} b_k^*$, $\sum_{i\in\omega_k} I_i p_i$, μ_k and $\bar{p}_k$*. Fortunately, *the summations of a variable within a neighboring window can be calculated using the integral images construct* [5], a.k.a. the summed area table. The integral images are calculated using dynamic programming with $\mathcal{O}(N)$ computational complexity irrespective of the window size r. Therefore, each iteration in our algorithm has $\mathcal{O}(N)$ computational complexity irrespective of the value of r.

A naive calculation of $\mathbf{Ap}$ has a computational complexity and memory of $\mathcal{O}(N^2)$ as well as much higher memory cost. An improvement over the naive method used in prior work on colorization [7] exploits the sparse structure of $\mathbf{A}$, and reduces the computation and memory cost of computing $\mathbf{Ap}$, both becoming proportional to $(2r+1)^2$ because total number of non-zero elements in $\mathbf{A}$ is $(2r+1)^2N$. Aggregated over the N pixels in the images, prior approaches for colorization in this optimization formulations have a per iteration cost of $\mathcal{O}(N(2r+1)^2)$. *Neither the computation nor the memory requirement of our method depends on r, and the per iteration cost is $\mathcal{O}(N)$. The lack of dependence on r also allows us to use large value of r to accelerate convergence of the iterations.*

For $r = 1$, the computation cost for each iteration in standard linear system solvers is comparable to our method. In these cases, *our method still offers significant computation and memory saving* because large values of r allow our method converge quicker and because we do not require the $\mathcal{O}(N(2r+1)^2)$ memory to represent the sparse matrix $\mathbf{A}$ for the prior implementations. Specifically, the conventional methods requires $\mathcal{O}(N(2r+1)^2)$ storage elements for representing the matrix $\mathbf{A}$, which are entirely eliminated in our algorithm.

To further accelerate the colorization process, we use a coarse-to-fine framework. We first obtain an estimate $\tilde{\mathbf{I}}_{\mathsf{sub}}$ for a subsampled image using the subsampled color scribbles. The estimation in the subsampled image also produces the coefficients $(\mathbf{a}_{\mathsf{sub}}, \mathbf{b}_{\mathsf{sub}})$ defined in (2). Next, $(\mathbf{a}_{\mathsf{sub}}, \mathbf{b}_{\mathsf{sub}})$ are upsampled to the original image size to obtain $(\mathbf{a}_{\mathsf{up}}, \mathbf{b}_{\mathsf{up}})$, and an initial guess for the $\mathbf{I}$ channel is calculated by $\tilde{\mathbf{I}}_{\mathsf{init}} = \mathbf{a}_{\mathsf{up}}\mathbf{Y} + \mathbf{b}_{\mathsf{up}}$. The Lanczos filters [11] are used in the downsampling and upsampling process to avoid aliasing. In the coarse level, we use a larger window radius r_{sub} to accelerate convergence of the conjugate gradient algorithm. In the fine level, a smaller window radius r_{up} is used to improve the accuracy of colorization since the linear model (2) is more accurate for smaller r. The estimate $\tilde{\mathbf{I}}_{\mathsf{init}}$ at the coarse level serves as a good initial guess, therefore accelerates the convergence of the refinement step.

4. RESULTS

We evaluate the proposed algorithm and benchmark it against the previously published algorithm for optimization-based colorization [1]; comparing the computation and memory requirements and the resulting colorized images. Because the method for defining the user specified colors for the scribbles is not the focus of our (or prior) work, this process is simplified and partly automated by beginning with a color image, which is converted to grayscale to obtain the input monochrome image. The user provides scribbles by marking up corresponding regions of the image using a suitable tool, specifically in our experiments we use the paintbrush tool in the GIMP [12] toolkit. The individual scribbles are then identified by using connected component analysis and for each specified scribble, a single user specified color value is obtained as the most saturated (having the largest mean squared sum for the I and Q chrominance channels) color in this region in the original image.

*the average value in a local window is obtained from the summation after normalization.

Parameters		Time (secs)		Memory (MB)	
		Proposed	Direct	Proposed	Direct
Children 265×320	$r = 1$	2.24	6.62	32	92
	$r = 2$	2.19	6.65	33	220
Hats 512×768	$r = 1$	18.4	40.3	160	791
	$r = 2$	18.3	39.9	165	1970
Woman 2560×2048	$r = 1$	237	681	2071	12502
	$r = 2$	233	-	2072	-

Table 1. Comparison of computation time and memory requirement for the proposed colorization algorithm against the conventional optimization based approach [1]. Estimates of memory are approximate, see accompanying text for details.

The proposed colorization algorithm is implemented in a coarse-to-fine framework as described in Section 3, where the coarse level uses a downsampling factor of 2 along each dimension and a window size $r_{\mathsf{sub}} = \frac{\min(w,h)}{2\times 50}$. At the fine level, we explore different values for the window size parameter r. For both levels, we set the regularization parameter $\epsilon = 10^{-5}$ and $\lambda = 10^3$, and terminate the iterations when either residual per pixel goes below a threshold of $\tau = 10^{-4}$ or when a maximum of 50 iterations are completed. The conventional optimization-based colorization approach [1], which uses a direct solver for the sparse linear system (6) and the proposed method are both implemented in Matlab[†].

Table 1 summarizes the run times required and estimated memory[‡] requirements for three differently-sized images used in our experiments. The proposed algorithm achieves a significant speed-up over the conventional optimization-based colorization approach. In particular, when using $r = 1$, the *woman* image with a size of 2560×2048 pixels, requires 237 seconds to colorize using our proposed algorithm in contrast with 681 seconds for the conventional optimization based approach. The run time requirements for the conventional approach are approximately $2.5\times$ those for the proposed method, consistently across the different image sizes.

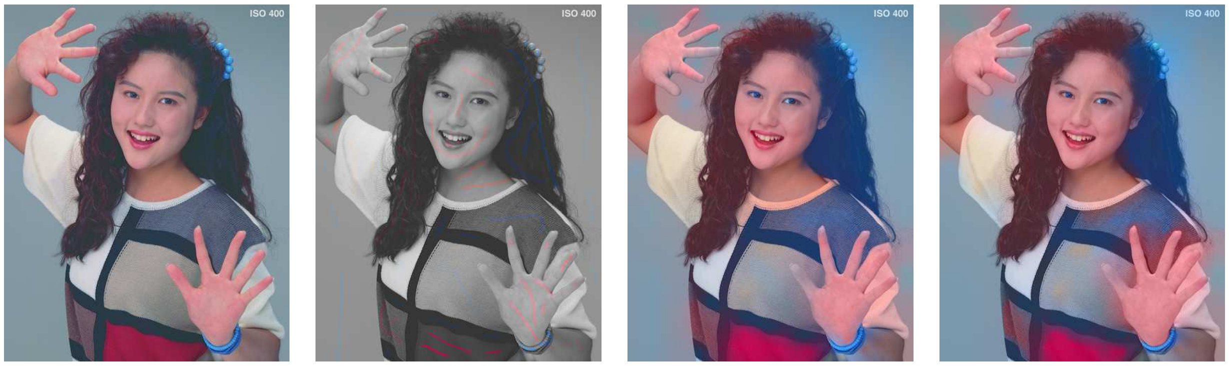
The reduction in memory requirement is even more significant and, importantly, grows with increase in the size of the image being colorized. For the 2560×2048 pixel *woman* image, with $r = 1$ the proposed algorithm uses 2071 Megabytes (MB) as opposed to 12502 MB required for the conventional optimization algorithm. For these 2560×2048 pixel images, with $r = 2$, the computation time and memory for the proposed algorithm remain approximately the same as those for the $r = 1$ cases, whereas the conventional approach fails to complete because the memory requirement exceeds the available main memory of 6GB, resulting in disk thrashing. The corresponding values for time and memory are therefore indicated by "$-$" in Table 1. This particularly illustrates the benefit of the proposed method: the significant reduction in memory (and time) requirements for the proposed method allows commodity hardware to be used for colorizing larger-sized high-resolution images encountered in realistic applications.

We also compared the output colorized images from our proposed approach with those obtained with the conventional optimization based approach and established that the approaches offer visually identical results. Three examples are shown in Figs 1, 2, and 3 which compare the colorization results using the proposed algorithm against those obtained with the conventional optimization based approach for three different input images labeled *children*, *hats* and *woman*. For all three examples, the colorized images appear are very similar for the proposed and conventional approaches: the colorized versions of *children* and *hats* images appear natural and artifact free where as the colorized *woman* image exhibits similar artifacts for both approaches. These artifacts can be overcome by using more color scribbles as inputs, which would raise the computational complexity of the conventional approach, though not for the proposed method. Thus the fact that that the computational complexity is largely independent of the number of color scribbles is also beneficial in practice. These examples and results on other images indicate that *our proposed approach provides visually identical results while requiring significantly less memory and computation time.*

We also empirically examined the convergence of the proposed algorithm. The mean squared residual $\left\|(\mathbf{L} + \lambda\mathbf{D})\tilde{\mathbf{I}} - \lambda\mathbf{I}_s\right\|^2 /N$ for the linear system (6) evaluated for the (current) iterate $\tilde{\mathbf{I}}$ in Algorithm 1 is plotted as a

[†]For the conventional approach, the Matlab backslash operator is used to invoke the direct solver, which is based on the highly optimized LAPACK package [13]

[‡]Unfortunately, the Matlab scripting environment does not directly provide an accurate estimate of memory requirements. The memory usage for the colorization process is therefore estimated as the difference between the peak memory requirements during the colorization process and the peak memory requirements for an idle Matlab process, both of which are estimated under Linux by running the "top" utility every 0.1 second and recording the peak value.



(a) Original color image (b) Input to the proposed algo (c) Colorization (proposed) (d) Colorization [1]

Figure 3. Colorization for a large image with sparse color scribble input.

function of the iteration number in Fig. 4 in two separate plots for the coarse and fine levels[§]. In these plots, in order to examine the convergence, we have deliberately used a larger number of iterations than required. We see that the algorithm exhibits the correct behavior; the mean squared residual converges toward 0 with increasing iterations. Additionally, we make the important observation that the convergence for the coarse level is faster because it uses a larger value of r. This behavior is consistent with our expectation that the coarse level estimate enables fast estimation of a good initial estimate.

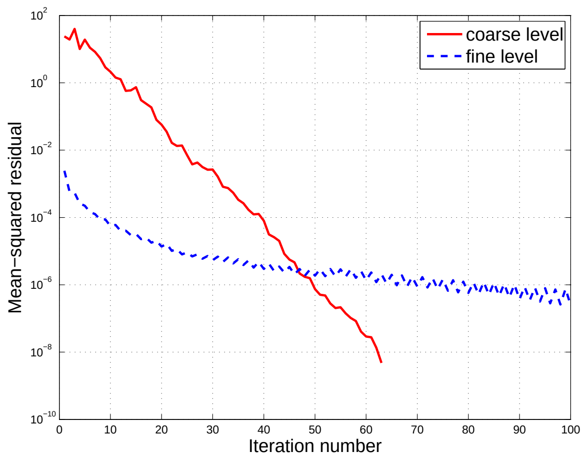


Figure 4. The mean-squared residue in (6) for the iterative estimate $\tilde{\mathbf{I}}$ in Algorithm 1 as a function of iteration count. Separate plots are shown for the coarse level and the fine level, identified by the legends. The specific plots correspond to the example of Fig. 2, other examples exhibit similar behavior.

5. CONCLUSION

In this paper, we develop a computationally efficient algorithm for image colorization that significantly reduces the computation and memory requirements compared with prior implementations of optimization-based colorization. The efficiency improvements result from three innovations: avoiding explicit construction of the large sparse matrix defining the linear system for the optimal solution, using integral images along with dynamic programming to reduce repetitive computation during the iterations required for obtaining the solution, and adopting a multiscale approach that first colorizes a coarse smaller version of the image from which information is propagated to the finer scale to speed up

[§]The residue at the first iteration in the fine level does not match that in the last iteration at the coarse level because different values of r are used in these cases and because upsampling alters the underlying image sizes.

convergence. For colorization of an image with N pixels using a linear model with window size $(2r+1)$ along each dimension, our method requires only $\mathcal{O}(N)$ computations per iteration compared with the $\mathcal{O}(N(2r+1)^2)$ computations required in prior implementations and also requires fewer iterations. The computation and memory savings allow us to perform the optimization-based colorization on typical commodity hardware for larger image sizes where the prior implementations become challenging due to their heavier demands for computation and memory. The algorithm we propose also has the advantage that the memory and computation cost are (largely) independent of the texture content in the image and of the number of regions for which the user specifies the approximate color via scribbles. In contrast, for prior algorithms developed for this problem, the computation/memory requirements increase with user inputs or amount of image texture and are therefore less predictable.

REFERENCES

1. A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM Trans. on Graphics*, vol. 23, no. 3. ACM, 2004, pp. 689–694.
2. B. Wang, Y. Yu, T. Wong, C. Chen, and Y. Xu, "Data-driven image color theme enhancement," in *ACM Trans. on Graphics*, vol. 29, no. 6. ACM, 2010, p. 146.
3. L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE Trans. Image Proc.*, vol. 15, no. 5, pp. 1120–1129, 2006.
4. K. T. Jiahao Pang, Oscar C. Au and Y. Guo, "Image colorization using sparse representation," in *Proc. IEEE Intl. Conf. Acoustics Speech and Sig. Proc.*, Vancouver, Canada, May 2013, pp. 1578–1582.
5. F. Crow, "Summed-area tables for texture mapping," *Computer Graphics*, vol. 18, no. 3, pp. 207–212, 1984.
6. C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A perceptually motivated online benchmark for image matting," in *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.*, 2009, pp. 1826–1833.
7. A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 30, no. 2, pp. 228–242, 2008.
8. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 1992.
9. Y. Saad, *Iterative methods for sparse linear systems.* Society for Industrial and Applied Mathematics, 2003.
10. K. He, J. Sun, and X. Tang, "Fast matting using large kernel matting Laplacian matrices," in *IEEE Intl. Conf. Comp. Vision, and Pattern Recog.* IEEE, 2010, pp. 2165–2172.
11. C. E. Duchon, "Lanczos filtering in one and two dimensions," *Journal of Applied Meteorology*, vol. 18, no. 8, pp. 1016–1022, 1979.
12. "The GNU image manipulation program, availabe at http://www.gimp.org/."
13. T. A. Davis, *Direct Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematic., 2006.