# Fast Random Walker with Priors using Precomputation for Interactive Medical Image Segmentation

Shawn Andrews, Ghassan Hamarneh, Ahmed Saad

Medical Image Analysis Lab, Simon Fraser University, Canada,
{sda56, hamarneh, aasaad}@sfu.ca

**Abstract.** Updating segmentation results in real-time based on repeated user input is a reliable way to guarantee accuracy, paramount in medical imaging applications, while making efficient use of an expert's time. The random walker algorithm with priors is a robust method able to find a globally optimal probabilistic segmentation with an intuitive method for user input. However, like many other segmentation algorithms, it can be too slow for real-time user interaction. We propose a speedup to this popular algorithm based on offline precomputation, taking advantage of the time images are stored on servers prior to an analysis session. Our results demonstrate the benefits of our approach. For example, the segmentations found by the original random walker and by our new precomputation method for a given 3D image have a Dice's similarity coefficient of $0.975$, yet our method runs in $1/25^{th}$ of the time.

## 1 Introduction

Segmentation is a crucial task in medical imaging. Manual segmentation by an expert is accurate, but is also very time consuming, while fully automatic and accurate segmentation techniques are not yet a reality, thus semi-automatic techniques become a necessity. While many semi-automatic techniques assume only user initialization, repeated user interaction is necessary to guarantee the accuracy required for medical imaging. Therefore, it is critical to speed up these techniques, especially in 3D, in order to minimize the time spent waiting between a user inputing information and seeing the results [1, 2].

A full survey of semi-automatic algorithms is beyond the scope of this work [3, 4]. At a high level, semi-automatic algorithms can be divided into several classes. One class involves the specification of an approximate boundary, which evolves towards the correct segmentation by minimizing a cost function derived from shape priors and image information [5, 6]. Another class of algorithms requires the user to specify sequential points on or near the boundary, and then the boundary is filled in between these points using a minimal path approach [7, 8]. A third class of algorithms asks the user to provide seeds, or pixels within specific regions, and then uses these seeds as a basis for the segmentation [9, 10].

An example of the last class of algorithms is the seeded random walker (RW_SD) [10], which is a graph-based approach to image segmentation that, along with its extensions, has garnered hundreds of citations in only a few years. It boasts many advantages, including weak boundary detection, robustness to noise, trivial generalization to simultaneous multi-region and 3D segmentations, a globally optimal solution ensuring

repeatability, a probabilistic segmentation that can be very useful in directing a user to areas of uncertainty [11], and the straightforward user input method of providing seed pixels - all important features in medical imaging. On the downside, once seeds are given, RW_SD computes the segmentation by solving a large system of equations, which can be slow. In [1], this problem is alleviated by introducing RW with precomputation (RW_PREC). Since medical images usually exist "offline" on servers for some time before they are segmented, some precomputation can be done before user input that allows a fast approximation to the segmentation once seeds are given, or "online". The speedup RW_PREC provides allows for user interaction with RW_SD in real-time.

Unfortunately, RW_SD has some limitations, specifically the segmentation is calculated based only on localized image data and disconnected regions must be seeded individually. These problems are addressed in [12], where regional intensity priors are introduced into the formulation. The priors result in more accurate segmentations and the ability to segment disjoint regions easily. However, in RW with priors (RW_PR), the image graph is not completely known offline, since priors are usually derived from the seeds and precomputation must be performed before seeds are given. This obsoletes the methods introduced in [1], which require the graph to be known. An algorithm with the robustness of RW_PR and the online speedup of RW_PREC would be a very useful interactive segmentation tool.

In this paper, we make the following contributions. First, starting with the random walker equations from [12], we derive an offline precomputation and an online approximation that allows for a significant online speedup that can be used in conjunction with priors. Secondly, we derive some additional precomputations that are performed offline to further speed up the online segmentation. Combining the robustness of RW_PR and the online speedup of RW_PREC, we create a useful interactive segmentation tool applicable to a more general class of problems than RW_PREC.

## 2 Methods: Random Walker Improvements

### 2.1 Random Walker Review

We begin by giving a review of existing RW algorithms for later reference. In the following derivations, we consider binary segmentations, but we note that our methods extend trivially to multiple labels, just as all previous RW algorithms discussed do. RW_SD [10] constructs a graph $G$ from an image where each vertex $v_i$ maps to a pixel $p_i$ and an edge $\{v_i, v_j\}$ implies $p_i$ and $p_j$ are neighboring pixels. For 2D images, the vertices will have degree $4$ or $8$, and for 3D images, the vertices can have degree from 6 to 26. The weight $w_{ij}$ of an edge is a function of the intensity difference between the pixels $p_i$ and $p_j$, with edges between similar pixels being assigned very large weight. The degree of a vertex $v_i$, $d_i$, is defined as the sum of the weights of all incident edges.

RW_SD then defines $L$ as the graph's Laplacian matrix, specifically if $L_{ij}$ is the $(i,j)^{th}$ component of $L$, then

$$L = \begin{cases} d_i & \text{if } i = j \\ -w_{ij} & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} . \quad (1)$$

The segmentation is performed by finding $x$, a vector of the probabilities that each node belongs to the object being segmented. We define $N$, $S$, $U = N - S >> S$ as the numbers of nodes in the graph, seeded nodes, and unseeded nodes, respectively. Following [10], once seeds are given we divide $L$ and $x$ into components corresponding to the seeded and unseeded nodes

$$L = \begin{bmatrix} L_S & B \\ B^T & L_U \end{bmatrix} \tag{2}$$

$$x = \begin{bmatrix} x_S \\ x_U \end{bmatrix} . \tag{3}$$

Here we use the convention of representing components of matrices and vectors composed of rows and/or columns corresponding to seeded nodes by a subscript $S$. Similarily, we represent components corresponding to unseeded nodes by a subscript $U$. It is shown in [10] that, since $x_S$ can be derived directly from the seeds, $x_U$ can be solved for by the linear equation

$$L_U x_U = -B^T x_S . \tag{4}$$

This is an equation of size $O(U)$, but due to the sparsity of the image graph, the matrix $L_U$ is sparse, and it can be solved in $O(U)$ time.

In RW_PR [12], $\lambda$ is introduced as a vector of the prior probabilities for each node, weighted by a scalar $\gamma$. Once seeds are given and the prior probabilities derived, the Laplacian is augmented with auxillary nodes. Dividing $L$ into components corresponding to the seeded, unseeded, and auxillary nodes, $L$ becomes

$$L = \begin{bmatrix} L_S & B & 0 \\ B^T & L_U + \gamma I_U & -\gamma\lambda \\ 0 & -\gamma\lambda & \gamma I_U \end{bmatrix} . \tag{5}$$

With the new image graph, (4) becomes

$$(L_U + \gamma I_U)\, x_U = -B^T x_S + \gamma\lambda . \tag{6}$$

This equation can also be solved in $O(U)$ time, and reduces to (4) when $\gamma = 0$.

In [1], it is shown how we can perform computations before seeds are given to speed up the solving of (4). Following [1], we define $\Lambda$ to be a diagonal matrix of $L$'s $K$ smallest eigenvalues and $Q$ to be a matrix who's columns are the $K$ corresponding eigenvectors. As $L$ is known before seeds are given, the expensive calculations required to find the eigenvalue/eigenvector pairs can be performed offline. The eigenvalue decomposition of $L$ is

$$L = Q\Lambda Q^T . \tag{7}$$

Though (7) is really an approximation, we can make it as close to exact as desired by increasing $K$. We define $g$ to be the eigenvector of $L$ with corresponding eigenvalue $0$ and

$$E = Q\Lambda^{-1} Q^T \tag{8}$$

to be the pseudoinverse of $L$ with the 0 on the diagonal of $\Lambda$ simply set to 0 in $\Lambda^{-1}$. We note that $g$ is a constant vector since $L$'s columns sum to 0. This gives

$$EL = (I - gg^T) \; . \tag{9}$$

We define $f$ by

$$Lx = f \tag{10}$$

for some $f$. If we knew $f$, we could approximate $x$ using the pseudoinverse by

$$ELx \approx (I - gg^T)x \approx Ef \; . \tag{11}$$

Once again, we divide $E$, $f$, and $g$ into components corresponding to seeded and unseeded nodes,

$$E = \begin{bmatrix} E_S & R \\ R^T & E_U \end{bmatrix} \tag{12}$$

$$f = \begin{bmatrix} f_S \\ f_U \end{bmatrix} \tag{13}$$

$$g = \begin{bmatrix} g_S \\ g_U \end{bmatrix} \; . \tag{14}$$

It can be shown that $f_U = 0$, so we want to find $f_S$. (10) and (11) give

$$L_S x_S + B x_U = f_s \tag{15}$$

$$-g_U g_S^T x_S + x_U - g_U g_U^T x_U = R^T f_S \; . \tag{16}$$

Taking (15) - $B$(16) gives

$$(L_S + B g_U g_S^T) x_S + B g_U g_U^T x_U = (I - BR^T) f_S \; . \tag{17}$$

We let $P = (I - BR^T)$, $\alpha = g_U^T x_U$, and $f_S = \hat{f}_S + \bar{f}_S \alpha$, and realize that

$$\begin{aligned} (L_S + B g_U g_S^T) &= (L_S - L_S g_S g_S^T) \\ &= L_S (I_S - g_S g_S^T) \\ &\approx L_S \; , \end{aligned} \tag{18}$$

as $g_S g_S^T$ is a constant matrix with entries $\frac{S}{N} << 1$ and contributes negligibly. So (17) becomes

$$L_S x_S + B g_U \alpha = P f_S \; . \tag{19}$$

Letting $f_S = \hat{f}_S + \bar{f}_S \alpha$, (19) can be divide into two equations

$$P \hat{f}_S = L_S x_S \tag{20}$$

$$P \bar{f}_S = B g_U \; . \tag{21}$$

We solve for $\alpha$ from (10) to get

$$\alpha = -\frac{g_S^T \hat{f}_S}{g_S^T \bar{f}_S} \ , \tag{22}$$

and finally (16) gives

$$x_U \approx R^T f_S + g_U \alpha \ . \tag{23}$$

Thus using the method from [1], solving (4), an equation of size $O(U)$, is reduced to the much quicker task of solving two equations of size $O(S)$, (20) and (21). However when priors are added to the RW formulation, the graph is no longer known offline, and thus this precomputation method is rendered useless.

### 2.2 Precomputation with Priors

Following [1], our goal is to derive a method for using offline precomputation to speed up the online computation for the more general RW_PR, where the image graph is altered after seeds are given. We begin with (10) extended to include priors and divided into components corresponding to seeded, unseeded, and auxillary nodes

$$\begin{bmatrix} L_S & B & 0 \\ B^T & L_U + \gamma I_U & -\gamma\lambda \\ 0 & -\gamma\lambda & \gamma\Phi \end{bmatrix} \begin{bmatrix} x_S \\ x_U \\ x_R \end{bmatrix} = \begin{bmatrix} f_S \\ 0 \\ f_R \end{bmatrix} \ , \tag{24}$$

which is equivalent to

$$\begin{bmatrix} L_S & B \\ B^T & L_U \end{bmatrix} \begin{bmatrix} x_S \\ x_U \end{bmatrix} = \begin{bmatrix} f_S \\ \gamma(\lambda - x_U) \end{bmatrix} \ , \tag{25}$$

Since we are not concerned with $x_R$. Similar to (11), we multiply (25) by $E$ to get

$$(I - gg^T) \begin{bmatrix} x_S \\ x_U \end{bmatrix} = \begin{bmatrix} E_S & R \\ R^T & E_U \end{bmatrix} \begin{bmatrix} f_S \\ \gamma(\lambda - x_U) \end{bmatrix} \ . \tag{26}$$

Now (25) and (26) give us, respectively,

$$L_S x_S + B x_U = f_s \tag{15}$$

$$-g_U g_S^T x_S + x_U - g_U g_U^T x_U = R^T f_S + \gamma E_U(\lambda - x_U) \ . \tag{27}$$

Letting $\alpha = g_U^T x_U$ as before, rearranging, and dropping the negligible $\left(-g_U g_S^T x_S\right)$ we get

$$(I_U + \gamma E_U) x_U - g_U \alpha = R^T f_S + \gamma E_U \lambda \ . \tag{28}$$

When working without priors, we eliminated $x_U$ from (16) by multiplying it by $B$ and subtracting it from (15). Now, however, $x_U$ in (28) has a matrix coefficient and cannot be eliminated as easily. To proceed in our derivation, we will calculate $J_U^{-1} =$

$(I_U + \gamma E_U)^{-1}$. However, defining $J_U$ requires the seeds and inverting a matrix of size $U \times U$ is too expensive of an operation to perform during the online phase of the algorithm. We can solve this problem by defining $J$ to be the extension of $J_U$ to size $N$ given by

$$J = (I + \gamma E) \tag{29}$$
$$\approx (QQ^T + \gamma Q \Lambda^{-1} Q^T) \tag{30}$$
$$\approx Q(I_K + \gamma \Lambda^{-1}) Q^T \tag{31}$$
$$\Rightarrow J^{-1} \approx Q(I_K + \gamma \Lambda^{-1})^{-1} Q^T \tag{32}$$

since $QQ^T \approx I_N$, which implies that $Q_U Q_U^T \approx I_U$. Thus

$$Q_U(I_K + \gamma \Lambda^{-1})^{-1} Q_U^T J_U \approx I$$
$$\Rightarrow J_U^{-1} \approx Q_U(I_K + \gamma \Lambda^{-1})^{-1} Q_U^T \; . \tag{33}$$

Since $(I_K + \gamma \Lambda^{-1})$ is a diagonal matrix with all positive entries, it is easily invertible online. We define $\hat{B} = B J_U^{-1}$. Taking (15) - $\hat{B}$(28) gives

$$L_S x_S + \alpha \hat{B} g_U = (I_S - \hat{B} R^T) f_s - \gamma \hat{B} E_U \lambda \; . \tag{34}$$

We take $f_s = \hat{f}_S + \bar{f}_S \alpha$ and $\hat{P} = (I_S - \hat{B} R^T)$ to get the equations

$$\hat{P} \hat{f}_S = L_S x_S + \gamma \hat{B} E_U \lambda \tag{35}$$
$$\hat{P} \bar{f}_S = \hat{B} g_U \; . \tag{36}$$

We solve for $\alpha$ using:

$$0 = g^T L x = \begin{bmatrix} g_S^T & g_U^T \end{bmatrix} \begin{bmatrix} f_S \\ \gamma(\lambda - x_U) \end{bmatrix} \tag{37}$$
$$g_S^T f_S = \gamma g_U^T (x_U - \lambda) \tag{38}$$
$$(g_S^T \bar{f}_S - \gamma)\alpha = -(g_S^T \hat{f}_S + \gamma g_U^T \lambda) \tag{39}$$
$$\alpha = \frac{g_S^T \hat{f}_S + \gamma g_U^T \lambda}{\gamma - g_S^T \bar{f}_S} \; . \tag{40}$$

Finally, now that we have $f_S$ we can plug it back into (28) to get

$$x_U = J_U^{-1}(g_U \alpha + R^T f_S + \gamma E_U \lambda) \; , \tag{41}$$

thus reducing the online solving of (6), an equation of size $O(U)$, to solving (35) and (36), two equations of size $O(S)$. This fulfills our goal of a RW formulation combining priors and precomputation (RW_PR_PREC).

### 2.3 Increased Speed using Extended Precomputation

By using the proposed RW_PR_PREC, we can greatly reduce the online runtime of RW_PR. However, due to the low connectivity of the image graph, $L_U$ is very sparse,

and (6) can be solved in $O(U)$ time. The online phase of RW_PR_PREC must take $O(U)$ time also, since it returns a probability vector of size $U$. We want to perform as few $O(U)$ cost computations as possible by minimizing the number of matrix multiplications between matrices of size $O(U)$. Analysis and optimizations of these asymptotic run times were not considered previously in [1], which we do here by analysis of matrix operations.

Currently, in the offline phase, we compute $Q$ and $\Lambda^{-1}$, but now we will precompute additional matrices to be used to speed up the online phase. The speedup will come from being able to retrieve the components of these matrices corresponding to the seeded nodes in $O(S)$ time. We note that $Q$ is an $N \times K$ matrix and for storage space considerations we do not want our precomputed matrices to be larger than that.

First we will look at computing

$$\hat{P} = (I_S - \hat{B}R^T) \tag{42}$$
$$= (I_S - BJ_U^{-1}Q_U\Lambda^{-1}Q_S^T) \tag{43}$$
$$= (I_S - (BQ_U)(I_K + \gamma\Lambda^{-1})^{-1}(Q_U^TQ_U)\Lambda^{-1}Q_S^T) \ . \tag{44}$$

This equation has 2 matrix multiplications between matrices of size $O(U)$, $BQ_U$ and $Q_U^TQ_U$, requiring $(SK + K^2)U$ scalar multiplications. Using

$$A_1 = LQ \tag{45}$$
$$A_2 = Q^TQ = Q_S^TQ_S + Q_U^TQ_U \tag{46}$$
$$A_3 = (I_K + \gamma\Lambda^{-1})^{-1} \tag{47}$$

(43) becomes

$$\hat{P} = (I_S - (A_{1S} - L_SQ_S)A_3(A_2 - Q_S^TQ_S)\Lambda^{-1}Q_S^T) \ , \tag{48}$$

where $A_{1S}$ is the $S \times K$ matrix of the rows of $A_1$ corresponding to the seeded nodes. Note $A_3$ doesn't eliminate any $O(U)$ operations, but is added for convenience. Every matrix in this equation has dimensions $S$ or $K$, thus we can calculate $\hat{P}$ without any $O(U)$ operations.

Next we will look at (36) for computing $\bar{f}_S$. The right hand side is

$$\hat{B}g_U = BQ_UA_3Q_U^Tg_U \ . \tag{49}$$

Without additional precomputation, this equation requires 1 matrix multiplication between matrices of size $O(U)$, $Q_U^Tg_U$, which takes $KU$ scalar multiplications. Note $BQ_U$ is not counted as it has already been computed and could be stored even without additional precomputation. Using

$$A_4 = Q^Tg = Q_S^Tg_S + Q_U^Tg_U \tag{50}$$

(36) becomes

$$\hat{P}\bar{f}_S = (A_{1S} - L_SQ_S)A_3(A_4 - Q_S^Tg_S) \ , \tag{51}$$

**Algorithm 1** OPT_RW_PR_PREC:

**Offline:**

Calculate $Q$ and $\Lambda^{-1}$ from $L$
$A_1 = LQ$
$A_2 = Q^T Q$
$A_3 = (I_K + \gamma\Lambda^{-1})^{-1}$
$A_4 = Q^T g$

**Online:**

$\hat{P} = (I_S - (A_{1S} - L_S Q_S)A_3(A_2 - Q_S^T Q_S)\Lambda^{-1}Q_S^T)$
$\hat{P}\hat{f}_S = L_S x_S + \gamma(A_{1S} - L_S Q_S)A_3(A_2 - Q_S^T Q_S)\Lambda^{-1}(Q_U^T\lambda)$
$\hat{P}\bar{f}_S = (A_{1S} - L_S Q_S)A_3(A_4 - Q_S^T g_S)$
$\alpha = \frac{g_S^T \hat{f}_S + \gamma g_U^T \lambda}{\gamma - g_S^T \bar{f}_S}$
$f_S = \hat{f}_S + \bar{f}_S\alpha$
$x_U = Q_U A_3((A_4 - Q_S^T g_S)\alpha + (A_2 - Q_S^T Q_S)\Lambda^{-1}(Q_S^T f_S + \gamma(Q_U^T\lambda)))$

which requires no $O(U)$ cost operations.

In the equation for $\hat{f}_S$, (35), we find the term

$$\hat{B}E_U\lambda = BQ_U A_3 Q_U^T Q_U \Lambda^{-1} Q_U^T \lambda \,. \tag{52}$$

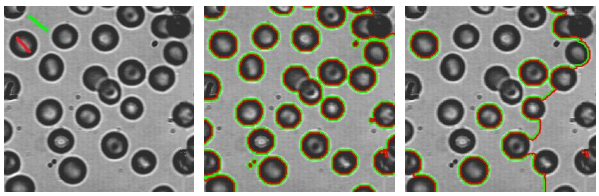
Both $BQ_U$ and $Q_U^T Q_U$ have been computed and could be stored. $Q_U^T\lambda$, however, requires data not available at the precomputation step, specifically the prior probabilities. Thus we must always compute $Q_U^T\lambda$ during the online phase.

The additional precomputations of $A_1$, $A_2$, $A_3$, and $A_4$ reduce the number of scalar multiplications greatly by $(SK + K + K^2)U$. A summary of the additional precomputations are in Algorithm 1, denoted OPT_RW_PR_PREC. We note that when not using priors, with RW_PREC, $A_1$ can still be used for additional speedup.

## 3 Results

We would like our results to show that using our precomputed data we make the RW online phase fast enough for interactive segmentation without compromising much accuracy. Therefore, we present results showing our high speed gains on real 2D and 3D data while maintaining negligible (and controlled) reduction in accuracy. We note that the speed increase allows much quicker seed editing and thus will translate to much improved accuracy per time spent by user.

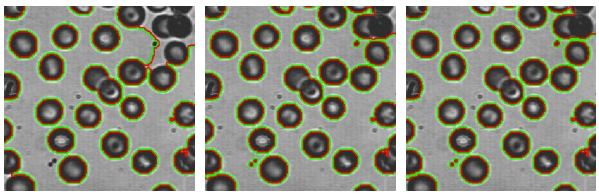
The experiments here were performed using unoptimized MATLAB code run on an Intel Core 2 Duo (2.4GHz) with 4GB of RAM. The algorithms were implemented by the authors, utilizing Grady's MATLAB Graph Toolbox [`http://www.cns.bu.edu/~lgrady`]. A negative exponential function was used for the edge weights, $w_{ij} = exp(-\beta(|i_a - i_b|))$, where $i_a$ is the intensity of pixel $a$. All experiment data was collected over 100 trials, and all parameters were chosen empirically and fixed across all compared methods. The only parameter effecting the speed of our method is $K$, the number of retained eigenvectors.



(a) Seeds

(b) Without Precomputation
$t_{on} = 0.552$ s
$t_{off} = 0$ s

(c) With Precomp.
$K = 20$
$Dice = 0.585$
$t_{on} = 0.043$ s
$t_{off} = 8.15$ s



(d) With Precomp.
$K = 40$
$Dice = 0.962$
$t_{on} = 0.063$ s
$t_{off} = 14.63$ s

(e) With Precomp.
$K = 80$
$Dice = 0.996$
$t_{on} = 0.097$ s
$t_{off} = 36.64$ s

(f) With Precomp.
$K = 160$
$Dice = 0.998$
$t_{on} = 0.178$ s
$t_{off} = 136.17$ s

Fig. 1: (color figure) Comparison of results with and without precomputation for segmentation using priors on an image of size $N \approx 72,000$ pixels. For $K$, the number of eigenvectors used, we report $Dice$, the Dice similarity coefficient between RW_PR's segmentation and OPT_RW_PR_PREC's segmentation, $t_{on}$, the online time taken, and $t_{off}$, the offline time taken. We note that we are only concerned with $t_{on}$ and with $K = 80$ our method achieves excellent results in less than a fifth of the time taken when not using precomputation. Red and green correspond to different region boundaries.

The accuracy of the segmentations generated by our algorithms are evaluated by their similarity to the segmentations generated by RW_PR; the accuracy of RW_PR is well justified in other works [12]. We note that the speed and accuracy of our algorithms depend on the image only through $K$, and while we leave analytical methods for finding optimal $K$ as future work, it was reported in [1] that $K = 40$–$80$ is often enough for 2D images, as our results in Section 3.1 corroborate, and we needed no more than $K = 350$ for larger 3D images as is seen in Section 3.2.

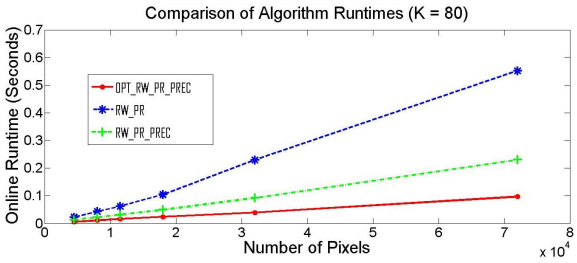



Fig. 2: Comparison of the runtimes of the original RW_PR (blue) and our proposed methods RW_PR_PREC (green) and OPT_RW_PR_PREC (red) for different resolutions of the image in Fig. 1. Note the standard deviation for the online runtimes are all under $0.05$ seconds.

The resolution and noisiness of an image affect how large of a $K$ is needed, and different noiseless images at the same resolution can require different values of $K$ to

be accurately segmented using precomputation. As seen in Fig. 1, even larger values of $K$ provide online runtimes much faster than can be achieved without precomputation, so choosing $K$ large enough to guarantee accuracy is our prime concern. As different images affect the speed and accuracy of our algorithms only through how large $K$ needs to be, and since we do not yet have an image dependent way to choose $K$ (except based on resolution and noise), results for a variety of images would be redundant. Thus we focus our results on single 2D and 3D images at varying resolutions and with varying levels of noise. We note that offline runtime increases with $K$, but does not affect the application to interactive segmentation.

### 3.1 2D Results

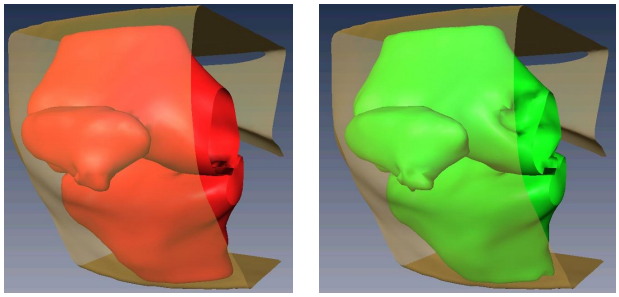
Tests were performed on the 2D image in Fig. 1 of size $N = 265 \times 272 \approx 72,000$ pixels with an 8-connected image graph, $\beta = 30$, $\gamma = 0.001$, and two regions, where one region was divided into multiple disconnected sections and seeds were only put in one of these sections. These segmentation times do not include calculating the priors, an efficient step which was performed online, and is similar in all cases. The priors were calculated using a non-parametric density estimation with a Gaussian kernel [12]. Fig. 1 shows the Dice similarity coefficient and the average runtimes in seconds for both the online and offline phases of OPT_RW_PR_PREC for different values of $K$ and compares the results to RW_PR, showing excellent speedup and minimal accuracy lost. Fig. 2 compares the runtimes of the different methods for different sized resolutions of the image in Fig. 1, again showing our precomputation gives excellent speedup.

### 3.2 3D Results

Tests were performed on a 3D CT image of the knee in Fig. 3 of size $N = 55 \times 55 \times 36 \approx 109,000$ voxels, a 26-connected image graph, and two regions, bone and non-bone. The bone region consists of 3 disconnected subregions (the femur, tibia, and patella). We tested the algorithms using priors by segmenting all the bones but placing seeds only in the tibia. We used RW_PR and OPT_RW_PR_PREC with $\beta = 100$ and $\gamma = 0.01$ and compared their average runtimes and the Dice similarity coefficient of their resulting segmentations. The average runtime of RW_PR was about $40.5$ seconds, and when $K = 350$ eigenvectors are used, the average runtime of OPT_RW_PR_PREC was about $1.56$ seconds. The Dice similarity coefficient between RW_PR's segmentation and OPT_RW_PR_PREC's segmentation was $0.975$. Thus our method achieved a speedup of $25$ times over RW_PR while maintaining excellent accuracy. The standard deviation of the runtimes of RW_PR was less than $1.0$ s, and the standard deviation of the runtimes of OPT_RW_PR_PREC was less than $0.1$ s.

### 3.3 Robustness to Noise

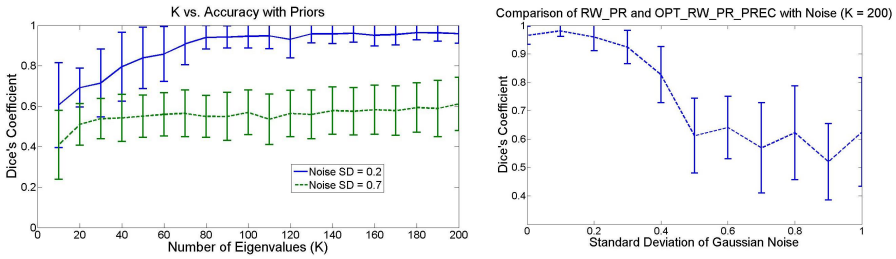
Here, we test the robustness to noise of OPT_RW_PR_PREC. We measured the similarity of the segmentations provided by the exact and approximate algorithms using the Dice similarity coefficient. The pixel intensities in our test images range from $0$ to $1$



(a) Without Precomputation (b) With Precomputation

Fig. 3: The bones of a knee segmented with $S = 100$ seeds in one of the bones. (a) was found using RW_PR in 40.5 seconds and (b) was found using OPT_RW_PR_PREC with $K = 350$ in 1.56 seconds. Dice's similarity coefficient between the two is $0.975$.

and various levels of Gaussian noise with standard deviations $\sigma \in [0, 1]$ were added to the 2D image in Fig. 1 of size $N = 265 \times 272 \approx 72,000$ pixels with an 8-connected image graph, $\beta = 30$, and $\gamma = 0.001$. From Fig. 4a we see that OPT_RW_PR_PREC still provides good segmentations for small amounts of noise up to $\sigma = 0.2$ (with Dice's similarity coefficient $> 0.95$) if a large enough $K$ is used. As the noise increases to $\sigma = 0.7$, Dice decreases. We can see the same trend in Fig. 4b, where $K = 200$ eigenvectors are used in the precomputation and the noise ranges from $\sigma = 0$ to $1$. We note that OPT_RW_PR_PREC can tolerate reasonable amounts of noise and still provide Dice similarity coefficient close to $1$.




(a) Dice vs. $K$ (b) Dice vs. Noise

Fig. 4: Effect of $K$ and noise on segmentation accuracy. (a) compares the Dice similarity coefficient between the segmentations found using RW_PR and OPT_RW_PR_PREC. Results are shown for two levels of noise and for multiple numbers of eigenvectors. (b) shows the Dice similarity coefficient between the segmentations at varying levels of noise with $K = 200$ eigenvectors and 20 trials for each level of noise. We see that large enough $K$ lets us account for reasonable amounts of noise.

## 4 Discussion and Conclusions

The above tests give some strong results. We see from Figs. 1 and 2 that the additional precomputation of OPT_RW_PR_PREC greatly outperforms RW_PR in 2D, achieving a segmentation in about one fifth of the time and with over $99\%$ similarity. From Fig. 3, we see the results are more pronounced in 3D, with OPT_RW_PR_PREC achieving speedups of a factor of $25$ over RW_PR while still finding an almost identical segmentation with over $97\%$ similarity. Furthermore, Fig. 2 shows that all the algorithms appear to increase linearly in runtime with the number of pixels, as predicted in Section 2.3.

Overall, we have derived a way of combining both precomputation and the use of priors into the popular RW algorithm. This allows RW to perform much faster segmentations when seeds and priors are either given or changed. Additionally, we've shown that some precomputations can be performed in addition to finding $Q$ and $\Lambda^{-1}$ that can greatly speed up the online phase of the algorithm. These improvements in speed provide a feasible way to enable the real-time editing of a wider variety of 2D and 3D images than was previously possible by allowing updating of both seeds and priors. This allows the user to ensure the accuracy of complex segmentations with minimal effort. Thus our contributions increase the usability and effectiveness of RW algorithms.

Future work will relate to using information from the image to automatically determine parameters for these algorithms. Specifically, the precomputation step introduces $K$, which needs to be set high enough to maintain accuracy. However, since the effect of a larger $K$ is seen mostly in the offline phase which doesn't effect interactivity, we currently simply err on the side of caution when selecting $K$.

## References

1. L. Grady and AK Sinop. Fast approximate random walker segmentation using eigenvector precomputation. In *IEEE Conf. CVPR*, pages 1–8, 2008.
2. Christopher J. Armstrong, Brian L. Price, and William A. Barrett. Interactive segmentation of image volumes with live surface. *Computers & Graphics*, 31(2):212–229, 2007.
3. S. D. Olabarriaga and A. W. M. Smeulders. Interaction in the segmentation of medical images: A survey. *Medical Image Analysis*, 5(2):127–142, 2001.
4. Y. Kang, K. Engelke, and W. A. Kalender. Interactive 3D editing tools for image segmentation. *Medical Image Analysis*, 8(1):35–46, 2004.
5. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int'l J on Computer Vision*, 1(4):321–331, 1987.
6. T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Trans. on Image Processing*, 10(2):266–277, 2001.
7. L. Cohen and R. Kimmel. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision*, 24:57–78, 1997.
8. W. Barrett and E. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1:331–341, 1997.
9. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-D images. *Proc. ICCV*, pages 105–112, 2001.
10. L. Grady. Random walks for image segmentation. *IEEE TPAMI*, 28(11):1768–1783, 2006.
11. A. Saad, G. Hamarneh, T. Moeller, and B. Smith. Kinetic modeling based probabilistic segmentation for molecular images. *Proc. of MICCAI*, 1:244–252, 2008.

12. L. Grady. Multilabel random walker image segmentation using prior models. *IEEE Conf. CVPR*, 1:763–770, June 2005.