

# REGULARIZED SEEDED REGION GROWING

R. BEARE

*CSIRO Mathematical and Information Sciences  
Locked Bag 17, North Ryde, Australia 1670  
Richard.Beare@csiro.au*

**Abstract** Region growing algorithms, including seeded region growing, are a widely employed technique used in image segmentation problems. It is common for region growing techniques to employ queue based techniques that sequentially add pixels to regions. These algorithms are very flexible, but it is sometimes desirable to be able to apply some constraints to the growing process that reflect some higher level knowledge about the problem.

This paper describes a technique that constrains (regularizes) the border smoothness of a growing region using a polygonal representation and provides some results illustrating a possible application. The technique can be usefully combined with traditional, unregularized techniques.

**Keywords:**

regularization, seeded region growing, polygon region representation.

## 1. Introduction

Region growing algorithms such as seeded region growing [1] and the watershed transform [5] are widely used to solve image segmentation problems. They begin with a set of markers or seeds which are grown until each image pixel is allocated to one marker. This growth is controlled by a priority queue mechanism which is dependent on the statistics of the image (or its gradient) covered by a region. The final shape of a region is not controlled by any higher level, problem dependent knowledge besides the marker selection. This can be both an advantage and a disadvantage. For example it may be desirable to extract regions with smooth boundaries. Existing region growing algorithms cannot guarantee this. Other segmentation techniques, such as those employing PDEs address some of these issues using very different approaches. This paper describes an extension to seeded region growing that allows constraints to be applied to the borders of regions.

## 2. Previous work

### 2.1 SEEDED REGION GROWING

The seeded region growing (SRG) method described in this section is a greedy algorithm, closely related to the watershed transform, that assigns a label to every pixel in the image while satisfying a connectivity constraint. The technique begins with a set of seeds that mark the regions to be segmented and uses a priority based system to grow regions one pixel at a time. This means that a single seed will grow to fill the entire image if there are no other seeds to compete with it. Other approaches place a threshold on the value of  $\delta$  (see below) and stop a region growing when the threshold is exceeded, so a single seed will not necessarily grow to fill the entire image. Regions can also be grown in an unseeded manner using split and merge or graph based approaches – see [4] for a review. Greedy approaches are parameter free, but require selection of seeds by some form of prefiltering or manual interaction. The regularization process described in this paper is applied to the greedy algorithm, although the same ideas could be applied to any region growing algorithm which operates on the borders of growing regions using priority based methods.

The following paragraphs give a more mathematically detailed description of seeded region growing.

SRG takes two images as input: a control image  $I$  and a seed image  $S$ .  $I$  can be virtually anything but in this case we will take the example of a single discrete gray-level image, on an 8-connected grid.  $S$  contains a collection of labelled binary regions,  $S_i \subset S, i \in 1, 2, 3, \dots$ . Regions with the same label do not have to be connected.

Let  $U$  be the set of pixels unassigned to any region but connected to at least one of them:  $U = x \in S, \forall i, x \notin S_i, N(x) \cap \cup_i S_i \neq \emptyset$ , where  $N(x)$  is the immediate neighborhood of  $x$ . At each step of the algorithm, all the points  $x \in U$  are examined in turn and a distance  $\delta(x, S_j)$  is computed between  $x$  and all the  $S_j$  it is neighboring, i.e:  $S_j, N(x) \cap S_j \neq \emptyset$ . The pixels  $y$  that possess the minimum distance are assigned to their neighboring region, and the process is repeated until all the pixels are associated with a region.

Typically, the distance  $\delta$  can be defined as follows:

$$\delta(x, S_j) = |I(x) - \text{mean}_{y \in S_j} (I(y))|,$$

where  $I(x)$  is the value in  $I$  at  $x$ .

### 2.2 LEVEL SET METHODS

Other segmentation techniques are able to include constraints on border smoothness. Methods using partial differential equations have been the subject of much investigation recently [6]. These methods minimize an energy function that can include terms related to boundary smoothness. Gradient descent approaches are used to find a local minimal surface that represents a segmentation of the image. PDE methods do not need the same sort of seed information that seeded region growing requires, which can be a significant advantage.

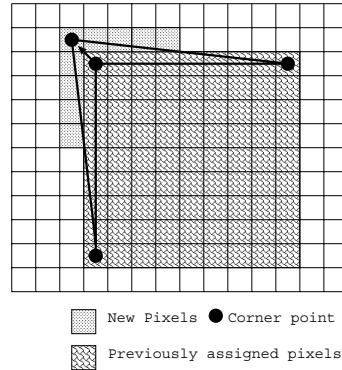


Figure 1. New pixels added to a region when a corner is moved

### 3. Regularized region growing

Regularised region growing permits a combination of both regularised and unregularized regions. Section 2.1 described the treatment of unregularized regions. Regularised regions are more complex.

The seed image  $S$  contains a collection of labelled binary regions  $S_i \subset S, i \in 1, 2, 3, \dots$ , as before. Each regularised region  $S_i$  is represented by a closed contour consisting of a set of corner points  $P_{i,1 \dots n}$  connected by straight line segments – i.e. a polygon. Point  $P_{i,k}$  is connected to  $P_{i,k-1}$  and  $P_{i,k+1}$  and  $P_{i,n}$  connects to  $P_{i,1}$ . The maximum distance between adjacent corners is the regularization factor defined by the user.

Each corner  $P_{i,k}$  may move to a number of new locations in the 4 or 8 connected local neighborhood. A move will increase the number of pixels belonging to a region (see Figure 1). The best move for a corner is the one with the minimum average pixel distance  $\delta_c = 1/m \sum_m \delta_j$ , where  $\delta_j$  is the distance measure used for pixels in unregularized region growing and  $m$  is the number of pixels labelled by a given move.

The best move is computed for each corner and placed in a priority queue with priorities  $\delta_c$ . This queue may also contain border pixels from unregularized regions.

The region growing then proceeds in the conventional fashion, with the highest priority elements being removed from the queue. If the element is a border pixel from an unregularized region it is processed in the conventional way (i.e. by adding neighbors). If it is a corner element from a regularised region then the relevant pixels are assigned to the region and the region mean is updated. The best new move is computed and the corner is re-queued. If the distance to a neighboring corner is greater than the user defined value, then the edge is split by adding a new corner at the midpoint. The best move for this corner is computed and placed on the queue.

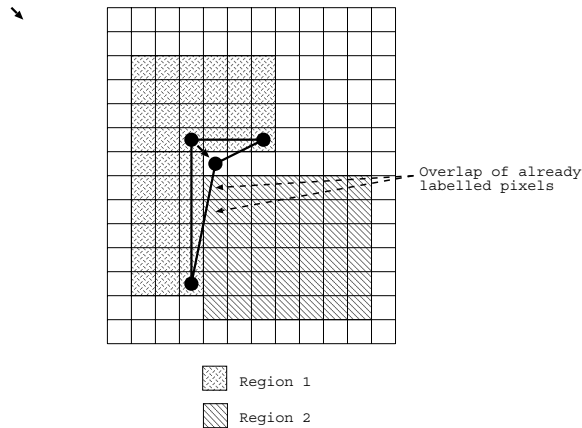


Figure 2. A move that is invalid due to overlap with already labelled pixels

## 4. Implementation Details

The regularization approach described above is reasonably simple. Unfortunately there are a number of complicating factors that need to be dealt with in order to produce a useful algorithm. These complications are caused by the pixel based representation of line segments and the desire to produce an algorithm that is strictly growing. The solutions presented in this section have been developed as the problems are understood, and are somewhat ad hoc.

Some simple optimization approaches are also described.

### 4.1 VALID DESTINATIONS FOR A CORNER “MOVE”

A critical part of the regularised region growing process is determining the best new position for a corner point. The new position will be a member of the 4 or 8 connected neighborhood of the corner, but not all members of this neighborhood are valid options. A location may be invalid for two reasons:

- Moving the corner to the location will attempt to add pixels that have already been assigned to another region (Figure 2).
- The location is not inside the projection of the two line segments connecting the corner to its neighboring corners (Figure 3). If locations outside this region are permitted, for example following track “A” in Figure 3, then the border of the region connecting neighboring corners will not remain straight – there will be a bend at the beginning of track “A”. It is possible to correct the problem by “unlabelling” some pixels, but this leads to potential stability problems with the algorithm. It is possible to efficiently test whether a neighboring pixel is inside the valid region.

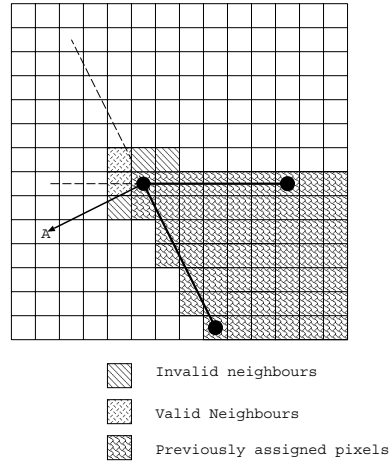


Figure 3. Pixel validity based on projection of region edges.

#### 4.2 MOVES THAT BECOME INVALID

A corner point will be queued with a priority value based on a specific move. It is possible that growth of other regions may have made the original move invalid, for the reason illustrated in Figure 2. It is therefore necessary to recheck the validity of a move. If a move has become invalid the corner must wait until it is re-queued through movement of one of its neighboring corners (see below).

#### 4.3 “LANDLOCKED” POINTS

When a corner is moved to a new location and the associated pixels are labelled the situation illustrated in Figure 4 may occur. The discrete nature of the seed image means that a corner point may become “landlocked” as one of its neighboring corners moves. If the corner point moved in Figure 4 remains the highest priority corner and keeps moving in the direction indicated then point A will become more landlocked. When it is eventually processed there will be no unlabelled pixels in the local neighborhood, meaning that there are no valid locations for the next move. This is not desirable because the growing process in the area around a landlocked corner will be halted. Landlocked corner points can be freed by moving them along either of the line segments connecting them to their neighboring corners until they have an empty neighboring pixel. This should only be done when a corner is landlocked by its own region, not when it collides with another region.

#### 4.4 RE-QUEUEING CORNER POINTS

If there are no valid locations in the neighborhood of a corner point then that corner will not be re-queued because there is no priority value associated with it. As neighboring corner points move it is possible that some of the previously

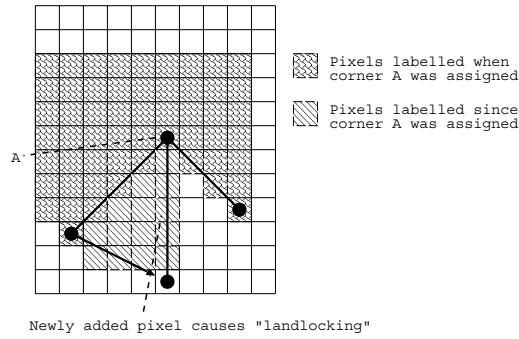


Figure 4. A corner point becomes landlocked as a region grows

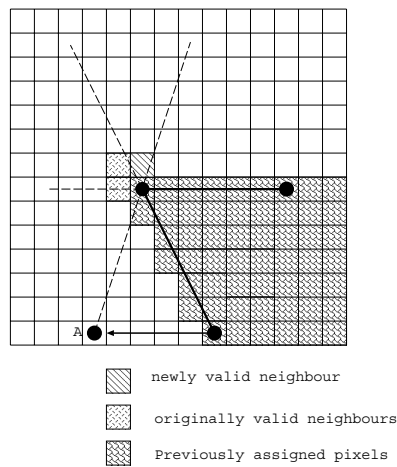


Figure 5. New locations become valid as adjacent corners move.

invalid locations will become valid (see Figure 5). Therefore, when a corner point is moved its neighboring corner points should be checked. If a neighboring corner is not on the priority queue then it should be re-examined to test whether any of the neighboring pixels now constitute a valid move. If a valid move now exists for that corner point it should be inserted into the priority queue.

#### 4.5 BORDER REPRESENTATION

The maximum distance between adjacent corner points is specified by the user. This means that it is possible to pre-compute all Bresenham lines that may form part of the boundary of a growing region. The line segments are stored as arrays of offsets and can be used to efficiently compute the priority of a corner point move. The ability to pre-compute the Bresenham lines is a useful optimization.

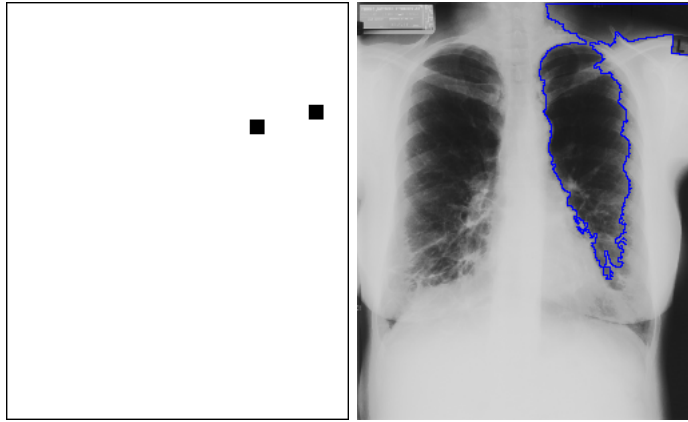


Figure 6. Initial seeds and results of unregularized region growing.

#### 4.6 APPROXIMATIONS

The original seeded region growing algorithm [1] employed an approximation to improve performance. The pixel distance is computed when the pixel is placed on the priority queue and never changed. The distance may have changed by the time the pixel is removed from the queue if the region has grown in other directions. It is possible to avoid this approximation with reasonable efficiency [2], but in most practical examples the differences are very minor. The algorithm described here employs a similar approximation, however the fact that more pixels may be involved in a single queued event and that some pixels may be involved in the calculation of priorities for adjacent corners could make the approximations more significant in some situations. This is yet to be investigated.

### 5. Results

The regularised algorithm may be used on its own or in conjunction with the unregularized version. When an image is segmented using only regularised seeds there are likely to be many unassigned pixels in the area where the regularised regions meet. A common reason for using regularised regions is to prevent a region growing through a small gap. In these situations it may be useful to use regularised regions for “foreground” objects and unregularized regions for background ones.

Figure 6 illustrates the segmentation of a lung in an x-ray image using unregularized region growing. The region grows through a gap near the collar bone and merges with the background.

If the lung is segmented using a regularised region while the background is unregularized (Figure 7) then the leak is prevented. In this case the maximum edge segment length is 20 pixels.

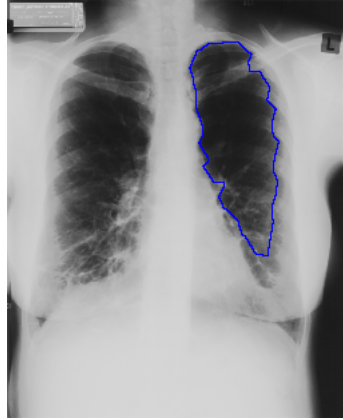


Figure 7. Results with regularised lung region and unregularized background using the seeds from 6.

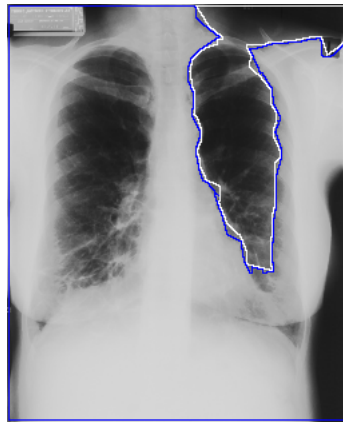


Figure 8. Results with regularised lung and background using the seeds from 6.

When both the foreground and background are regularised (Figure 8) the leakage reappears. In this example the collarbone is very narrow so a regularised background region is unable to grow along it. Unassigned pixels appear at the interface between the two regularised regions.

## 6. Performance

The algorithm has been implemented in C++ using the priority queues provided with the standard template library. Performance for unregularized seeds alone is lower (approximately 25%) than an implementation using more specialized data structures such as those described in [3]. Running time does not seem to change measurably when using regularised regions or a mixture of reg-

ularised and unregularized ones. This indicates that the increased complexity of the polygonal representation is offset by the reduction in queue size.

Execution time for the lung images ( $256 \times 313$ ) shown in this paper on a 1GHz Pentium III running Linux is approximately 0.15 seconds.

## 7. Conclusions and further work

This paper has described a method for efficient regularization of a well known region growing algorithm. The approach uses a polygon to represent the growing region. The regularised method can be usefully combined with the traditional approach.

Further work on this topic includes weighting the priority of a move by the angle between the connecting line segments, application to the watershed transform, simplification of the rules governing the growth of the polygon and improvements in the efficiency of the implementation using improved queuing data structures. Another approach that may be of interest is to represent the regularised region using a bezier curve instead of a polygon.

## References

- [1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:641–647, 1994.
- [2] R. Beare and H. Talbot. Exact seeded region growing for image segmentation. In *Digital Image Computing: Techniques and Applications*, pages 132–137, December 1999.
- [3] E. Breen and D. Monro. An evaluation of priority queues for mathematical morphology. In J. Serra and P. Soille, editors, *Mathematical morphology and its applications to image processing*, pages 249–256. Kluwer, 1994.
- [4] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *CVGIP*, 29:100–132, 1985.
- [5] F. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, September 1990.
- [6] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.