19th Computer Vision Winter Workshop
Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5, 2014

jSLIC: superpixels in ImageJ

Jiří Borovec and Jan Kybic

Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech republic
jiri.borovec@fel.cvut.cz

Abstract *This paper presents the implementation and particular improvements on the superpixel clustering algorithm - SLIC (Simple Linear Iterative Clustering). The main contribution of the jSLIC is a significant speed-up of the original clustering method, transforming the compactness parameter such that the value is image independent, and a new post-processing step (after clustering) which now gives more reliable superpixels - the newly established segments are more homogeneous. The improvements of speed and quality are shown on real images. We implemented the new jSLIC in Java and made the source code publicly available. Also we created a plug-in in ImageJ/Fiji which is commonly used as a research and development platform in biology and medical imaging.*

1 Introduction

The amount of data in medical imaging to be processed is increasing - images in histology can easily have 50.000×50.000 pixels or even more. The segmentation or registration of these large images is very demanding. The complexity of segmentation and registration can be reduced by using superpixels [7, 4].

In the past, several superpixel algorithms were introduced which were based on, for example the watershed approach, level-set based geometric flow, mode-seeking segmentation scheme or graph-based (a comparison is presented in [2, 9]). Recently, SLIC (Simple Linear Iterative Clustering) [2] was introduced for general images and presented as a powerful intermediate phase for further image segmentation, classification and registration.

We chose SLIC because of its universality and linear (and low) complexity (see Sec. 1.1). This fact is important for pre-processing large images. SLIC has a high rate in boundary recall and a low rate of under-segmentation error [2]. Another benefit is the low number of parameters to be set and an opportunity to influence the size and compactness of the resulting superpixels.

The main contribution of this work is a significant speed-up over the original clustering method and also providing a multi-thread version. Moreover, the regularisation parameter is transformed into the range $\langle 0, 1 \rangle$ to be more image independent. We also propose a new post-processing step which gives more reasonable superpixel shapes even for larger superpixel grid spacing.

While ImageJ [1, 5, 11] (and Fiji, derivation of ImageJ) is commonly used as a research and development platform in medical imaging, there is no implementation of superpixels. We decided to implement SLIC in Java and call it **jSLIC**. The Java source code and a ready to install Fiji plug-in are publicly available[1].

In Sec. 1.1, we briefly introduce the general SLIC algorithm. Then, we discuss the implementation and proposed speeds-ups together with the explanation and gain of each partial procedure in Sec. 2. Later in Sec. 3, we speak about the post-processing phase where we define the problem, summarize the existing approach and introduce ours and present the differences on the atcome using both methods.

1.1 SLIC superpixels

SLIC [2] is an adaptation of the k-means [6] algorithm for superpixel generation with two important distinctions: (a) the weighted distance measure

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2} \tag{1}$$

combines colour d_c (using the CIELAB colour space, which is widely considered as perceptually uniform for small colour distances) and spatial proximity d_s and (b) the search space is reduced by limiting to a region $2S \times 2S$, proportional to the superpixel size S. The search space reduction has a great impact on the speed of whole algorithm, resulting on a complexity of only $O(N)$ instead of $O(NkI)$ for standard k-means, where N is the number of pixels in a image, k is the number of clusters and I is the number of iterations [3].

2 Implementation and speed-ups

Several implementations of SLIC already exist. The author of [2] provides a C source code[2] which was wrapped into Python[3] (we use this code as the reference of SLIC). Other implementations can be found also for Matlab (VLFeat[4] library). For real-time computer vision problems, SLIC has been also transformed to be fully processed on graphic cards with some minor improvements as gSLIC [10].

[1] http://fiji.sc/CMP-BIA_tools
[2] http://ivrg.epfl.ch/research/superpixels
[3] https://github.com/amueller/slic-python
[4] http://www.vlfeat.org/

Figure 1: Sample image - Lena (image size 512×512 pixels) clustered by the original SLIC (middle) and our jSLIC (right) method. You can see that most of the superpixels are equal except those around Lena's eyes where jSLIC added extra superpixels for the white which we consider as the right choice.

We implement the plug-in in Java with maximal focus on compatibility with ImageJ. We used the ImageJ API and as much as possible we had to use the native Java structures a few times to keep the clustering process as fast as possible.

2.1 Regularisation constant

SLIC contains a regularisation parameter f which influences the compactness of clustered superpixels. This constant f weights the space distance d_s and it is expressed as $f = \left(\frac{m}{S}\right)^2$ from eq. (1) where (according to the notation in [2]) S is the initial superpixel size and m is a parameter related to maximal colour distance N_c in the range $(0, \infty)$. We propose instead to use a parameter r defined in the range $\langle 0, 1\rangle$, where 0 means the minimal and 1 the maximal compactness.

$$f = S \cdot r^2 \tag{2}$$

In our experiments we found that the optimal default regularisation value $r = 0.2$ works well for most cases. It is a good compromise between the superpixel compactness and fitting boundaries of the expected object in image.

2.2 Using Look-Up Tables

We analysed the possibility of using precomputed Look-Up Tables (LUTs) to avoid repetitive computing of the same distances d_s in eq. (1) or converting the same colours again. We found that we can achieve significant speed-up in specific cases (especially for colour conversion) mentioned below.

Spatial distance in regular grid. The metric used in SLIC clustering contains a proximity distance

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

where $[x_i, y_i]$ and $[x_j, y_j]$ are coordinates of the cluster centre and a pixel respectively. In a regular image grid, these distances are the same for all cluster centres and its proportional subset of neighbouring pixels. Using this precomputed distance LUT, we gain a 5% speed-up.

Colour conversion. Most commonly used images are in RGB colour space and we compute the colour distance in CIELAB colour space (see Sec. 1.1). It means that each image needs to be converted from RGB to CIELAB which is quite time consuming. We found that the number of used unique colours in images is usually smaller than the number of pixels in the image. Average images has at maximum 50% of unique colours/pixels (e.g. Lena with size 512×512 pixels). For medical images, the ratio is even smaller. For example, a common image of stained histological section (see [4]) contains less than 5% of unique colours/pixels. So we create the conversion LUT just when it is needed so each used colour is computed just once. It gives us a speed-up of about 60%.

method	speed-up
original SLIC	0%
jSLIC initial	26.6%
spatial proximity LUT	33.7%
colour conversion LUT	217.3%
jSLIC fast (distance & colour)	264.9%
jSLIC parallel (4 threads)	495.4%

Table 1: Table presents the speed-ups of each proposed procedure. All following ratios are mean speed-ups evaluated over several histological images with different image size (see Fig. 5) and they express the relative speed-up to the original SLIC. On the beginning the jSLIC (implementation according [2]) is about 27% faster then the original SLIC implemented in C. Later the pre-computation of distances and converting each colour just once brings 5% and 58% speed-up respectively comparing to the initial jSLIC and about 64% both together. In the end, the parallelisation for 4-threads gives another speed-up of 37% to the fast jSLIC.

2.3 Multi-threading

As the clustering is computed locally (for each superpixel only its proportional $2S \times 2S$ area, see Sec. 1.1), is quite simple to split the process by subsets of superpixels and/or image blocks into independent threads in both phases (assignment and update).

We apply the parallelism usually on the main loop in the given phase - in the assignment phase each thread takes only a subset of all superpixels/clusters, and the update is computed per image blocks, such that each thread processes one image block.

We perform this parallelisation on a computer with 8-

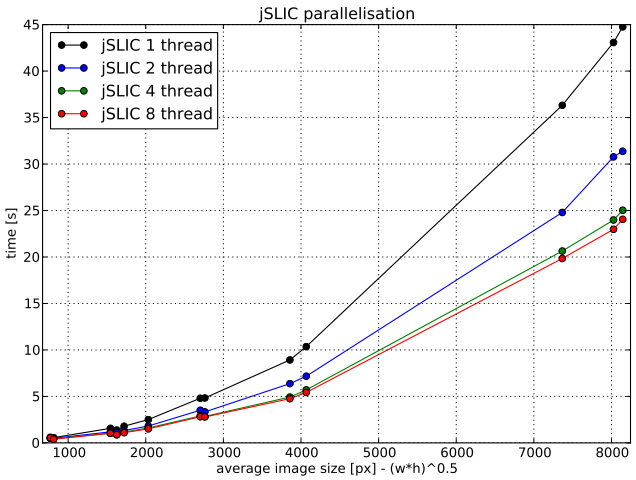


Figure 2: We ran a benchmark on several histological images with different image size and parallelism on $1-8$ threads. You can see that with the increasing number of used threads the processing time also decrease. The most significant speed-up is between the single and 4-thread version.

cores and the results are presented in Fig. 2. The most significant speed-up is between the single and 4-thread version. You can see that the 8-thread version for small images takes even more time which is due to multi-threading overhead.

3 Post-processing of outliers

The SLIC clustering generates a quite large number of unconnected components (small regions which belong to a superpixel but they are not connected to it). The number of unconnected regions depends on superpixel compactness but in average (for regularisation $r = 0.2$) there are about $3M$ unconnected regions where $M = \frac{w \cdot h}{S^2}$ is number of expected superpixels depending on image size and initial superpixel size S.

At first, all connected components c_i have to be found. We use a region growing method to compute all independent components c_i (assuming 4-neighbour). Then, for each component c_i we find a set of neighbouring components Ω_i. This early stage is the same for the original SLIC as it is for jSLIC post-processing.

Original SLIC post-processing [2]. The authors measure the relative area $c_i^s = \frac{|c_i|}{S^2}$ of each component and merge small components if $c_i^s < 0.25$. For relabelling they simply use the label c_i^* of the first component from Ω_i such that c_i^* is the neighbouring component of the first pixel belonging to c_i.

We found out that this simple approach is not sufficient (see Fig. 3), because some unconnected components are merged to a superpixels even it would be more reasonable to merge then into another neighbouring superpixel or introduce them as new superpixels. The author deals with this issue by estimating smaller superpixels [7] and setting the superpixel size smaller than the smallest detail in the image that they want to distinguish.

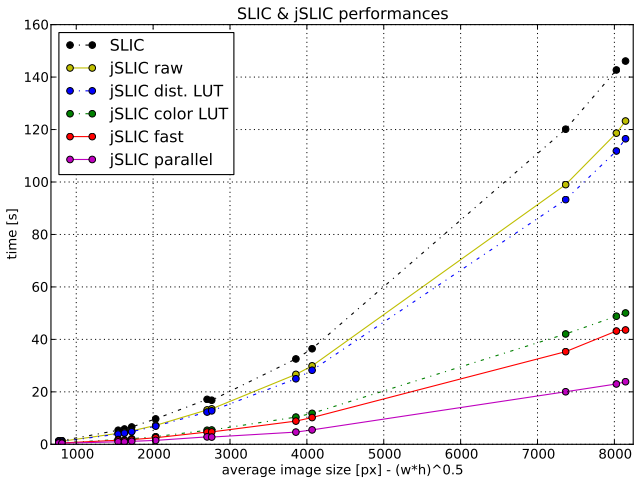


Figure 5: The chart presents the time dependency of complete superpixel clustering by SLIC and different variants of jSLIC depending on the number of pixels in the image. In average, the parallel jSLIC is 6 times faster than the original SLIC implementation.

Proposed jSLIC post-processing. We propose a different post-processing step which takes into account all surrounding components and their similarity by colour and area. We compute mean colours c_i^{lab} and relative area c_i^s for all components. Then, we find the most similar component c_i^* by computing the difference $l_i\,(c_j)$ between the colours of the components (3) and choosing the closest component (4) with minimal distance

$$l_i\,(c_j) = \frac{||c_i^{lab} - c_j^{lab}||_2}{c_j^s} \quad (3)$$

$$c_i^* = \arg\min_{c_j \in \Omega_i} (l_i\,(c_j)) \quad (4)$$

where the $||\cdot||_2$ is the Euclidean distance.

We experimented with the SLIC relabelling condition for unconnected components (see Fig. 3). We found the original $c_i^s < \epsilon$ condition insufficient even with various threshold values ϵ, because it does not take into account the colour similarity. We propose a condition which solves this problem - the unconnected regions are merged if

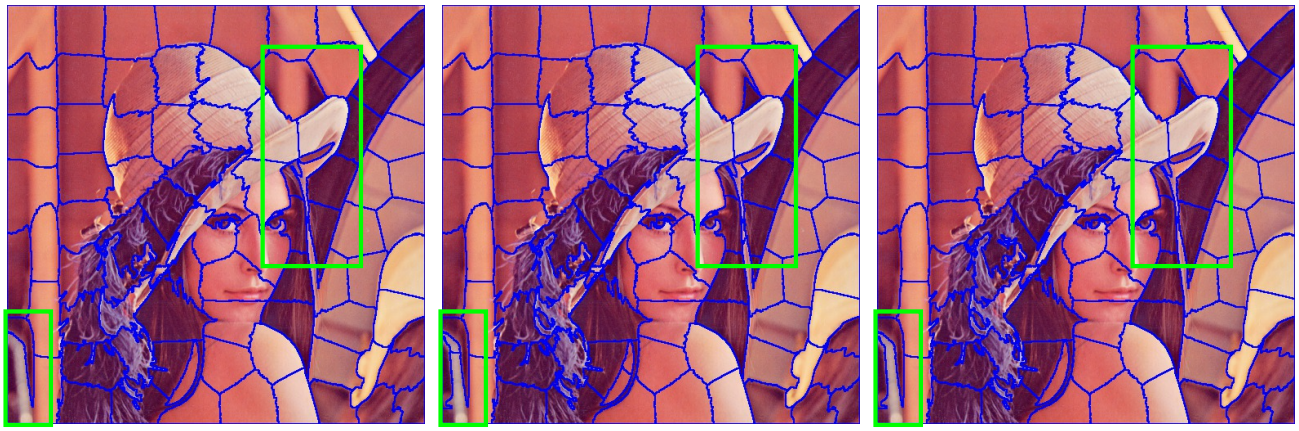
$$\left(\frac{c_i^s}{4}\right)^2 \cdot (1 + l_i\,(c_i^*)) < \varepsilon \quad (5)$$

where $\frac{c_i^s}{4}$ expresses the relative superpixel size to the maximal superpixel size $2S \times 2S$. Experimentally, we set the threshold $\varepsilon = 0.25$.

4 Comparison and discussion

We applied jSLIC on several histological images of various image sizes, up to about 8.000×8.000 pixels, on a standard computer with a 4-core processor and 8Gb RAM. As a reference we used the original SLIC implementation in C and compared it to our jSLIC in Java. The time dependency of all partial speed-ups on image size is presented in Fig. 5. In average, we found the parallel jSLIC to be 6 times faster than the original SLIC implementation.

The experiments with parallelism show that the jSLIC is optimal when using up to 4-threads. Using more threads due



(a) (b) (c)

Figure 3: We compared the original SLIC condition for merging unconnected components $c_i^s < \epsilon$ applying two different thresholds - original $\epsilon = 0.25$ in (a) and decreased $\epsilon = 0.06$ in (b). For all relabelling, we used our choosing of most similar neighbouring component c_i^* defined in eq. (4). In (c) we introduced also our condition for merging described in eq. (5). You can see that most of the superpixels in (a, ,b, c) are the same. The difference can be seen in the right upper part of the image. Original SLIC (a) just holds one large superpixel comparing to (b, c) which reasonably adds one more superpixel. On the other hand (b) adds some other small superpixels in nearly homogeneous areas, while (c) holds still single superpixels.

to the threading overhead, does not brings bigger improvements in performance.

For the evaluation of the proposed post-processing step, we used a few images from the Berkeley Segmentation Dataset [8] and some stained histological images (see Fig. 4). We made a visual evaluation of segmented superpixels with respect to the amount of detail extracted from a given image. For both methods we set the same configuration - the same initial superpixel size $S = 30$ and regularisation constant $r = 0.2$. To present the differences, we chose a detail in each image where the improvements can be easily seen (the rest of the image is usually segmented equally).

The advantage of the jSLIC post-processing is the ability to segment also smaller details than the initial superpixel size S in region it is needed and the ability to keep larger superpixels in more uniform image parts (see Sec. 3). We benefit from this fact when segmenting large histological images, where a big reduction of problem complexity is needed. For instance, have a look at the sample of histological image (Fig. 4 bottom), where the jSLIC is clearly capable of estimating the hole in the tissue comparing to original SLIC method.

5 Conclusion

We presented a Java-based open source implementation of jSLIC superpixel clustering with better performance than the original SLIC. Moreover, we proposed a different regularisation parameter, which influences the compactness of resulting superpixels and propose a default value $r = 0.2$. The new post-processing step gives more reliable superpixels shapes, with no need of decreasing superpixel size.

Acknowledgement

The authors were supported by The Czech Science Foundation under project P202/11/0111 and by The Grant Agency of the CTU Prague under project SGS12/190/OHK3/3T/13.

References

[1] M.D. Abramoff, P.J. Magalhães, and S.J. Ram. Image processing with ImageJ. *Biophotonics international*, 11(7):36–42, 2004.

[2] R. Achanta and A. Shaji. SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *Pattern Analysis and Machine Intelligence, IEEE*, 34(11):2274 – 2282, 2012.

[3] R. Achanta, A. Shaji, K. Smith, and A. Lucchi. Slic superpixels. Technical report, 2010.

[4] J. Borovec. Fully automatic segmentation of stained histological cuts. In Libor Husník, editor, *17th International Student Conference on Electrical Engineering*, pages 1–7, Prague, 2013. CTU in Prague.

[5] T.J. Collins. ImageJ for microscopy. *Biotechniques*, 43(S1):S25–S30, July 2007.

[6] J.A. Hartigan and M.A. Wong. Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, Oct. 1979.

[7] A. Lucchi, K. Smith, and R. Achanta. Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks With Learned Shape Features. *Medical Imaging, IEEE*, 31(2):474 – 486, 2012.

[8] D. Martin and C. Fowlkes. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision, IEEE*, number July, 2001.

[9] P. Neubert and P. Protzel. Superpixel Benchmark and Comparison. Technical report, 2012.

[10] C.Y. Ren and I. Reid. gSLIC: a real-time implementation of SLIC superpixel segmentation. Technical report, 2011.

[11] C. Schneider, W.S. Rasband, and K.W. Eliceiri. NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, 9(7):671–675, June 2012.

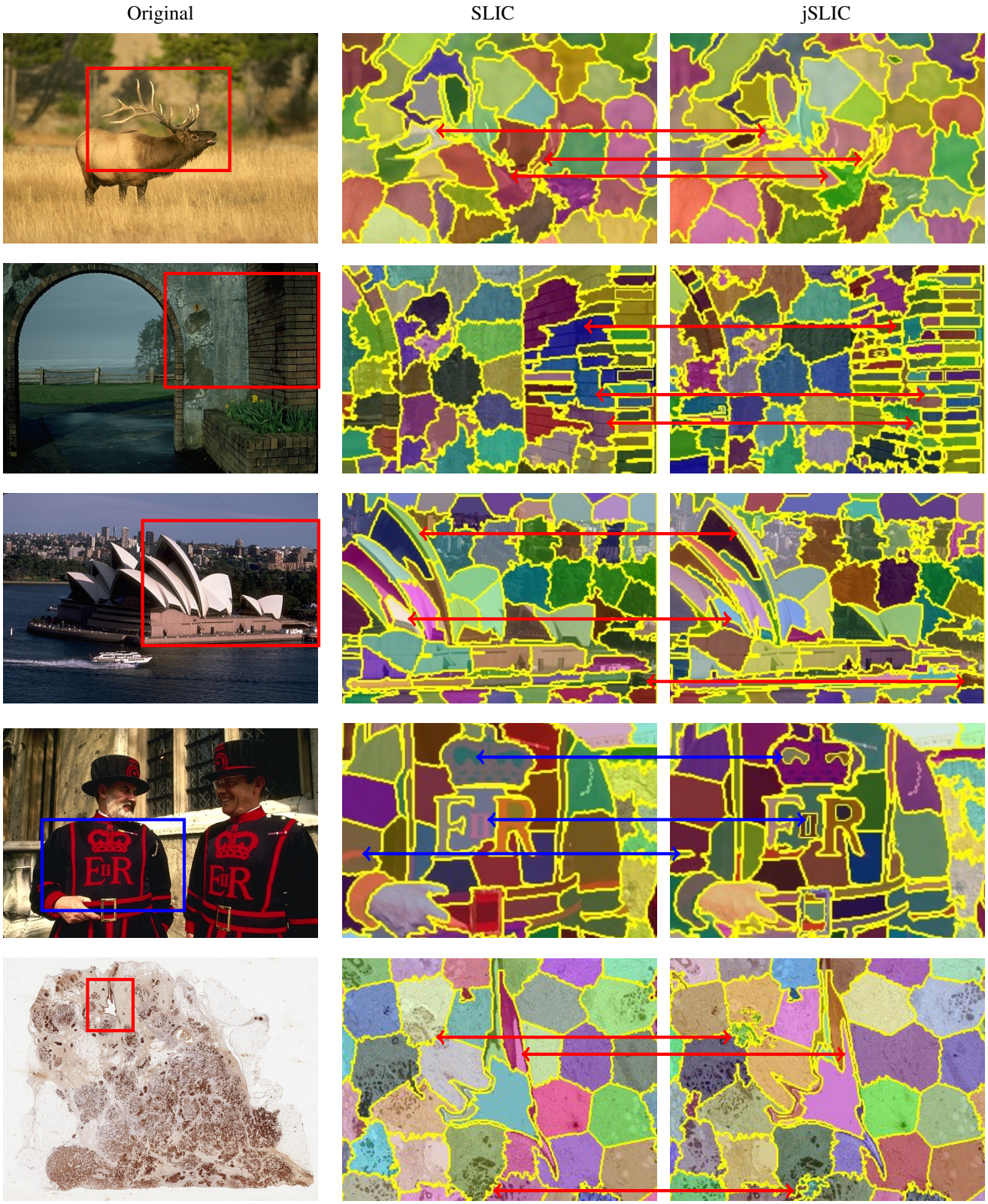


Figure 4: We run the original SLIC (middle) and jSLIC (right) on the Berkeley Segmentation Dataset [8] and some stained histological images using the same configuration for both. To present the differences we chose from each image only a part/detail where improvements can be easily seen (the rest of the image is usually segmented equally). The reason for more reliable superpixels by jSLIC is, because it takes into account all neighbouring connected components and their similarity by colour.