PARALLEL MEAN SHIFT ACCURACY AND PERFORMANCE TRADE-OFFS

Kirsty R. Duncan, Robert Stewart and Greg Michaelson

School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, United Kingdom

ABSTRACT

This paper decomposes the algorithmic parameters that affect the accuracy and parallel run times of mean shift segmentation. Following Comaniciu and Meer [1], rather than perform calculations in the feature space of the image, the joint spatial-range domain is represented by the image space, with feature space information associated with each point.

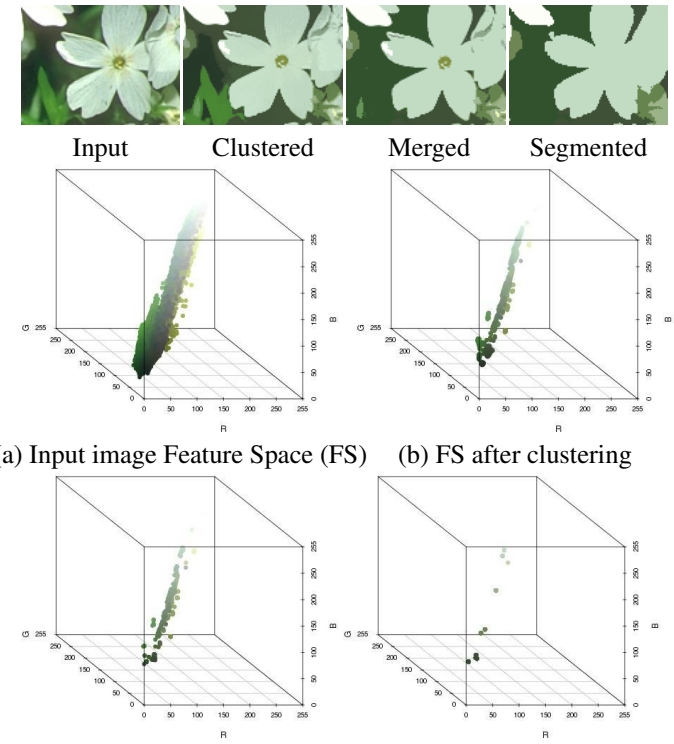
We report parallel speedup and segmentation accuracy using a standardised segmentation dataset and the Probabilistic Rand index (PRI) accuracy measure. Changes to the algorithmic parameters are analysed and a sweet spot between PRI and run time is found. Using a range window radius of 20, spatial window radius of 10 and threshold of 50, the PRI is improved by 0.17, an increase of 34% which is comparable to state of the art. Mean shift clustering run time is reduced by 97% with parallelism, a speedup of 32 on a 64-core CPU.

***Index Terms*—** Image segmentation, mean shift, algorithm analysis, parallel processing.

1. INTRODUCTION

The mean shift clustering algorithm is widely used in the computer vision domain but runs too slowly for certain applications [2]. Algorithm parameter tuning and parallel processing can alleviate this trade off by speeding up execution times, without comprising algorithmic robustness. Mean shift is proposed by Fukunaga and Hostetler [3] as a method to cluster data. It has applications in image processing: edge-preserving image filtering and segmentation (Comaniciu et al [1]), text detection in images (Kim et al [4]), real-time face tracking (G Bradski [5]) and real-time object tracking (Allen et al [6], Ramesh and Meer [7]).

A mean shift algorithmic optimisation in [2] uses a Hessian matrix to reduce the number of iterations to convergence. An implementation optimisation in [8] uses a binary tree structure to store neighbouring points in a feature space for efficient search. An implementation in [9] computes the peaks of point samples in parallel, and remaining points are assigned to their nearest peaks. Our algorithm is based on an approach in [1], by performing mean shift on the spatial-range domain where feature space information is associated with each point. The clustering phase is parallelised to reduce run time costs when the complexity of segmentation increases.



(a) Input image Feature Space (FS) (b) FS after clustering

(c) FS after clustering and merging (d) FS after segmentation

Fig. 1: Image and corresponding Feature Space (FS) at each stage of mean shift segmentation.

2. MEAN SHIFT SEGMENTATION

The approach is to tune mean shift algorithmic parameters and to use parallelism in the joint spatial-range domain to achieve high Probabilistic Rand Index (PRI) and visual segmentation accuracy, and high run time performance. We compute segmentation by performing clustering (Figure 1b), merging (Figure 1c) and then thresholding (Figure 1d).

2.1. Algorithmic Components

Clustering Each pixel in an image can be represented by three values in a colour-space. A common colour-space is RGB, where each pixel has a vector position according to its red, green and blue values. This feature space can be a probability density function of colour. Dense regions of this

space correspond to the local maxima of the probability density function. The cluster associated with each can be found.

The *multivariate kernel density estimator* $\hat{f}_{h,k}(\mathbf{x})$ gives an estimate for the density within a window h around a point x. The modes of the density estimator are found at the points with zero gradient $\bigtriangledown \hat{f}_{h,k}(\mathbf{x}) = 0$. The *mean-shift vector* $\mathbf{m}_{h,k'}(\mathbf{x})$ is the normalised gradient of the density estimator. It points in the direction of maximum increase in density within the window and can therefore be used to define a path towards the local maximum of the density estimate.

The *joint spatial-range representation* encodes 3D RGB values, known as the range, and also 2D XY values, corresponding to position in the image. Each cluster represents a connected object of a certain colour in the image, rather than separate segments of the same colour around the image [1]. The joint spatial-range *mean shift vector* is:

$$\mathbf{m}_{h_r,h_s,k'}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i^{r,s} k'(||\frac{\mathbf{x}^r - \mathbf{x}_i^r}{h_r}||^2) k'(||\frac{\mathbf{x}^s - \mathbf{x}_i^s}{h_s}||^2)}{\sum_{i=1}^{n} k'(||\frac{\mathbf{x}^r - \mathbf{x}_i^r}{h_r}||^2) k'(||\frac{\mathbf{x}^s - \mathbf{x}_i^s}{h_s}||^2)} - \mathbf{x}, \tag{1}$$

where $\mathbf{x}^{r,s}$ is the 5D position in the joint-space, $\mathbf{x}^r$ is the 3D component of $x^{r,s}$ corresponding to the range, with range window radius h_r and $\mathbf{x}^s$ is the 2D component corresponding to the position in the image, with spatial window radius h_s.

We use the *Epanechnikov kernel,* $k(x)$ with the property:

$$-k'(\frac{x}{h}) = \begin{cases} 1 & 0 \leq x \leq h \\ 0 & x > h. \end{cases} \tag{2}$$

The clustering algorithm works as follows. For each point in the feature space:

1. define the window of radius h around this point and calculate the mean shift vector from it;
2. if the mean shift vector is non-zero, move to the point that the vector indicates and go back to step 1;
3. if the mean-shift vector is zero at this new point, it is the peak of the original point.

All points with the same peak are merged in the output.

Cluster Merging and Thresholding After mean shift clustering, clusters of "peaks" have formed around the feature space of the image. In order to segment an image, each of these clusters should be considered as a single entity. To achieve this, all peaks within range and spatial window radius of each other are merged into one peak. If a region contains less than a threshold number of pixels, it is to be merged with the closest region of above-threshold size.

2.2. Algorithmic Complexity

The mean shift clustering algorithm operates on the feature space of an image, so the run time for calculation of the mean shift vector is proportional to the range window radius cubed. When clustering the image in the the joint spatial-range domain, the image space can be used for calculations, with range information attached to each point. The complexity is reduced to the spatial window radius squared.

We identify three parameters as having an effect on the run time of the algorithm as well as the accuracy of the output; 1) ***Range window size***, which is the radius of the spherical window in the three feature space dimensions of the joint space when computing the mean shift vector. The wider the radius, the greater the volume of points considered for calculation of the vector and the fewer clusters identified in the output of the clustering algorithm. 2) ***Spatial window size***, which is the radius of the circular window in the two image space dimensions of the joint space. The wider the radius, the greater the circular area of points considered. 3) ***Threshold***, which is the minimum number of pixels required to make up a single region. These affect the mean shift segmentation computational cost, and hence affect run time. Parallel processing of the mean shift kernel also affects run time, but will not affect segmentation accuracy.

3. EVALUATION

3.1. Measuring Algorithmic Accuracy

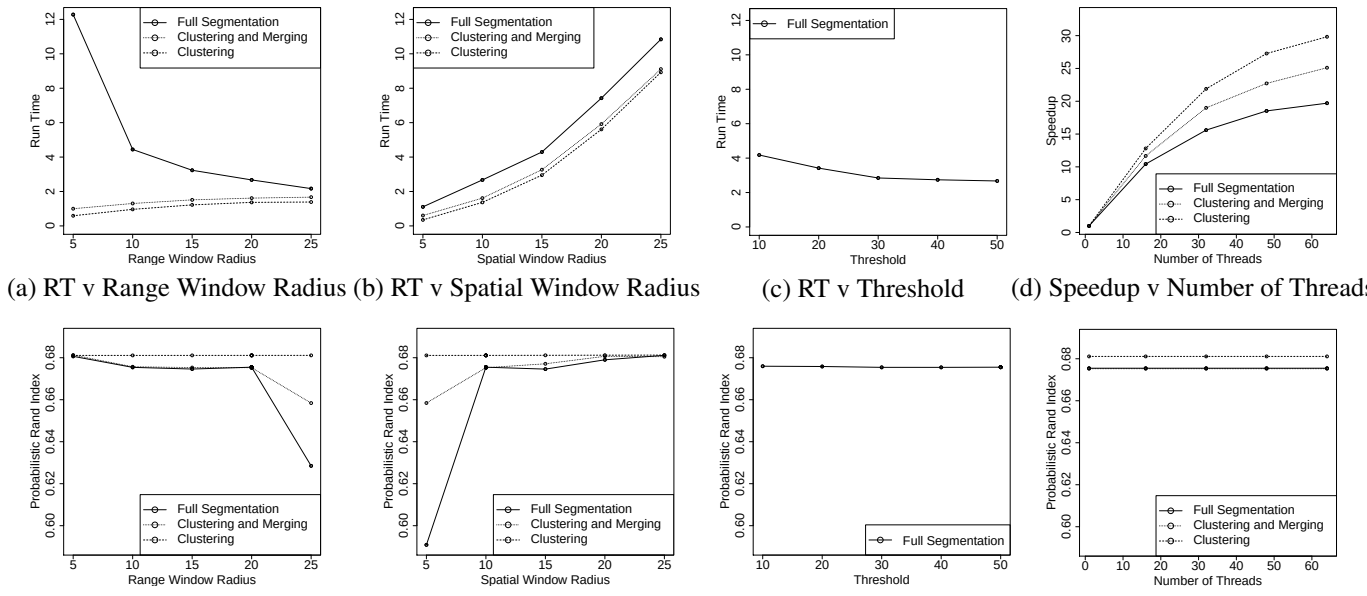
The PRI [10] is used to measure accuracy. It provides a method to compare algorithm output with hand segmented (ground truth) images. It accommodates refinement in regions that humans find ambiguous and penalises them elsewhere, by considering several ground-truth images for each image.

This index takes a value between 0 and 1, where 0 means the segmentation bares no resemblance to the ground truth segmentations and 1 means all segmentations are identical. The PRI measure is proposed in [10], which remarks that a PRI of 0.577 or less represents poor segmentation. A state of the art segmentation method [11] achieves a PRI value of 0.7.

3.2. Implementation

Our C++ implementation is available in an open access dataset [12]. Mean shift clustering is parallelised using OpenMP [13] on a 64-core AMD Opteron 1.4GHz CPU. The reported run times are the mean of five executions for each configuration of the mean shift segmentation parameters.

Run time and PRI are measured for spatial and range window radii of 5 to 25, thresholds from 10 to 50 and threads from 1 to 64. The *124084* image from the Berkeley Segmentation Dataset and Benchmark [14] is measured in Section 3.3. Four other 481x321 pixel natural images from this dataset have been processed with our implementation and both the accuracy and run time results are very similar so they are omitted for brevity. For each configuration of parameters, the PRI of the output image was calculated using the Matlab toolbox from [11].



(a) RT v Range Window Radius (b) RT v Spatial Window Radius (c) RT v Threshold (d) Speedup v Number of Threads

(e) PRI v Range Window Radius (f) PRI v Spatial Window Radius (g) PRI v Threshold (h) PRI v Number of Threads

Fig. 2: Run Time (RT) in seconds and Probabilistic Rand Index (PRI) Results for each parameter.

3.3. Results

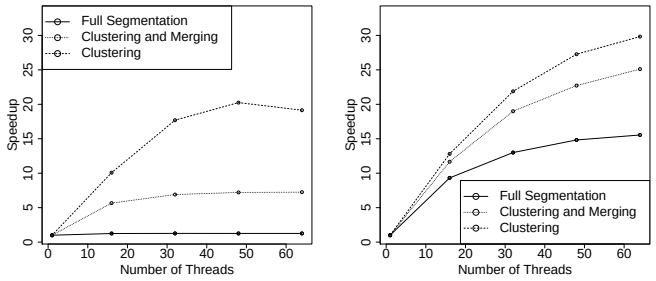
Range Window Size Figures 2a and 2e show run time and PRI versus range window size. The spatial window radius is 10 and the threshold is 50. As the range window radius increases, the PRI does not vary greatly; however the run time decreases, particularly between window radius 5 and 10. Using thresholding with small range window radius significantly increases run time. Clustering and merging with a small range window limits the impact of clustering, producing outputs with many small segments. Thresholding eliminates these, but with so few above-threshold segments to merge with, it merges large areas of the image into a single segment. A large range window increases computational costs of the clustering phase. A small range window increases the complexity of thresholding. A range window radius of 20 is desirable, as the PRI decreases significantly for larger radius, and the run time does not decrease by increasing the radius past 20.

Spatial Window Size Figures 2b and 2f show run time and PRI versus spatial window size. The range window radius is 20 and the threshold is 50. Increasing the spatial window size from 5 to 10 increases the PRI from 0.59 to 0.68. For spatial window sizes above 10, the PRI increases marginally. Run time increases with the square of the spatial window radius. This is expected as each calculation of the mean shift vector is proportional to the spatial window squared. Thus it is desirable to select a spatial window radius of 10, as the PRI does not increase substantially by increasing the radius above 10, but the run time increases dramatically.

Thresholding Figures 2c and 2g show the run time and PRI versus threshold. Here the spatial window radius is 10 and range window radius is 20. Increasing the threshold value does not have a noticeable effect on the PRI, but moderately decreases run time. As can be seen in Figures 4e and 4f, increasing the threshold decreases the number of small superfluous segments. A threshold of 50 increases performance without affecting the PRI, while improving the visual result.

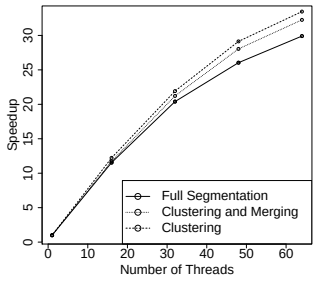
Parallelism Figures 2d and 2h show speedup and PRI versus number of threads. Speedup is T_1/T_n where T_1 is sequential runtime with one thread and T_n is runtime with n threads. Here the spatial window radius is 10, range window radius is 20 and the threshold is 50. Increasing the number of threads decreases the run time while the PRI accuracy is constant. The nature of cluster merging and thresholding make them prone to race conditions resulting in non-deterministic segmentation results, so have been implemented sequentially.

Figures 3a, 3b and 3c show the speedup for different configurations of the window radii. Speedup for full segmentation is significantly lower than for clustering and merging. When both window radii are small, as in 3a, thresholding dominates the run time, as seen in Figure 2a, and there is very little segmentation speedup. When both windows are relatively large, speedup is similar for clustering, clustering and merging and full segmentation, since the parallel clustering phase is the biggest contributor to the run time. However, the sequential run time for large windows, presented in 3d, is an order of magnitude larger than for small or medium windows. The benefit of parallelism for small and medium window radii

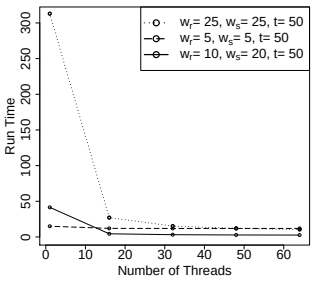


(a) Speedup v Threads 1

(b) Speedup v Threads 2



(c) Speedup v Threads 3



(d) Run time v Threads

Fig. 3: Speedup versus number of threads for each of 3 configurations of the window radii. (a) presents results for $w_r = 5, w_s = 5$, (b) for $w_r = 10, w_s = 20$ and (c) for $w_r = 25, w_s = 25$. Threshold is 50 for all full segmentation results. (d) presents full segmentation absolute run time versus number of threads for each configuration.

(a) Range window radius 5. 59 segments.

(b) Range window radius 25. 35 segments.

(c) Spatial window radius 5. 14 segments.

(d) Spatial window radius 25. 46 segments.

(e) Threshold 10. 117 segments.

(f) Threshold 50. 44 segments.

Fig. 4: Resulting images using different parameters. Unless stated otherwise, $w_s = 20$, $w_r = 20$ and threshold is 50.

is reduced, but the absolute time for large windows is much greater. For 16 threads or more, the window radii sweet spot identified as $w_r = 20, w_s = 10$ outperforms the other cases significantly. The three lines appear to have deceptively flat lined. The sweet spot has run time 3.2s, 2.8s and 2.7s for 32, 48 and 64 threads respectively. Segmentation with window radii $w_r = 25, w_s = 25$ has run time 15.3s, 12.0s and 10.5s respectively and segmentation with $w_r = 5, w_s = 5$ has run time 11.9s, 12.0s and 12.0s respectively.

The thresholding step is optional [15] and only helpful for certain applications, *e.g.* background subtraction. Considering the significant increase in run time, as illustrated in Figures 2a and 2b, and significant decrease in the accuracy shown in Figure 2e for large range window and Figure 2f for small spatial window, the added run time costs due to thresholding are only worthwhile in application specific contexts.

Visual Results Figure 4 displays the segmentation results for the highest and lowest values measured for each parameter. An outline of the identified segments overlays the original image. 4a and 4c show that for small window radii, the thresholding phase overcompensates due to the large number of small segments present after clustering and merging. Figures 4e and 4f show that thresholding has a subtle effect despite its significant run time; the main segments are not altered but small superfluous segments are removed.

4. CONCLUSION

This paper analyses the mean shift segmentation algorithm when performed on the image space of a colour image, rather than the feature space. Effects of varying the range and spatial window radii and the threshold on the run time and algorithmic accuracy were measured and a sweet spot between the two was identified. Implementing the algorithm in this way reduces its complexity and allows wider feature space windows to be used without a polynomial increase in run time. The PRI increased by 0.17, an increase of 34%, using the sweet spot identified as a range window radius of 20, spatial window radius of 10 and threshold of 50. This is comparable to state of the art in [11] for the Berkeley segmentation dataset. Parallelism of mean shift segmentation without thresholding decreased run time by 97%, a speedup of 32.

Acknowledgements We acknowledge the support of the Engineering and Physical Research Council, grant reference EP/N028201/1 (Border Patrol: Improving Smart Device Security through Type-Aware Systems Design).

5. REFERENCES

[1] Dorin Comaniciu and Peter Meer, "Mean shift analysis and applications," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. IEEE, 1999, vol. 2, pp. 1197–1203.

[2] Changjiang Yang, Ramani Duraiswami, Daniel DeMenthon, and Larry Davis, "Mean-shift analysis using quasinewton methods," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*. IEEE, 2003, vol. 2, pp. II–447.

[3] Keinosuke Fukunaga and Larry Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on information theory*, vol. 21, no. 1, pp. 32–40, 1975.

[4] Kwang In Kim, Keechul Jung, and Jin Hyung Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1631–1639, 2003.

[5] Gary R Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*. IEEE Computer Society, 1998, p. 214.

[6] John G Allen, Richard YD Xu, and Jesse S Jin, "Object tracking using camshift algorithm and multiple quantized feature spaces," in *Proceedings of the Pan-Sydney area workshop on Visual information processing*. Australian Computer Society, Inc., 2004, pp. 3–7.

[7] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. IEEE, 2000, vol. 2, pp. 142–149.

[8] Daniel DeMenthon and Remi Megret, *Spatio-temporal segmentation of video by hierarchical mean shift analysis*, Computer Vision Laboratory, Center for Automation Research, University of Maryland, 2002.

[9] Honggang Wang, Jide Zhao, Hongguang Li, and Jianguo Wang, "Parallel clustering algorithms for image processing on multi-core cpus," in *Computer Science and Software Engineering, 2008 International Conference on*. IEEE, 2008, vol. 3, pp. 450–453.

[10] Ranjith Unnikrishnan and Martial Hebert, "Measures of similarity," in *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*. IEEE, 2005, vol. 1, pp. 394–394.

[11] Allen Y Yang, John Wright, Yi Ma, and S Shankar Sastry, "Unsupervised segmentation of natural images via lossy data compression," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.

[12] Kirsty Duncan, Robert Stewart, and Greg Michaelson, "Open Access dataset for 'Parallel Mean Shift Accuracy and Performance Trade-Offs', accepted in IEEE ICIP 2018," May 2018, `http://doi.org/10.17861/fd3ee9dd-dd6d-47f2-8b22-12e634cb6556`.

[13] B. Chapman, G Jost, and R. van der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*, The MIT Press, 2007.

[14] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, July 2001, vol. 2, pp. 416–423.

[15] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.