ACCELERATION STRATEGIES FOR GAUSSIAN MEAN-SHIFT IMAGE SEGMENTATION.

Miguel Á. Carreira-Perpiñán.

Dept. of CSEE, OGI, Oregon Health & Science University.

1 Abstract

Gaussian mean-shift (GMS) is a clustering algorithm that has been shown to produce good image segmentations (where each pixel is represented as a feature vector with spatial and range components). GMS operates by defining a Gaussian kernel density estimate for the data and clustering together points that converge to the same mode under a fixed-point iterative scheme. However, the algorithm is slow, since its complexity is $\mathcal{O}(kN^2)$, where N is the number of pixels and k the average number of iterations per pixel. We study four acceleration strategies for GMS based on the spatial structure of images and on the fact that GMS is an expectation-maximisation (EM) algorithm: spatial discretisation, spatial neighbourhood, sparse EM and EM–Newton algorithm. We show that the spatial discretisation strategy can accelerate GMS by one to two orders of magnitude while achieving essentially the same segmentation; and that the other strategies attain speedups of less than an order of magnitude.

2 Gaussian mean-shift (GMS)

Given dataset $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^D$, define a Gaussian kernel density estimate with bandwidth σ:

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} K\left(\left\|\frac{\mathbf{x}-\mathbf{x}_n}{\sigma}\right\|^2\right) \qquad K(t) = e^{-t/2}$$

GMS is an iterative algorithm to find a mode of p:

- it has no step size and is an EM algorithm with global, linear convergence (Carreira-Perpiñán & Williams 03)
- clustering: $\mathbf{x}_n, \mathbf{x}_m$ in same cluster if they converge to same mode
- nonparametric clustering; σ determines the number of clusters
- popular in computer vision (segmentation, tracking; Comaniciu & Meer)
- based on Fukunaga & Hostetler '75 (also Cheng '95, Carreira-Perpiñán '00, Comaniciu & Meer '02, etc.)

for $n \in \{1,\dots,N\}$ — *For each data point*
  $\mathbf{x} \leftarrow \mathbf{x}_n$ — *Starting point*
  repeat — *Iteration loop*
    $\forall n\colon p(n|\mathbf{x}) \leftarrow \frac{\exp\left(-\frac{1}{2}\|(\mathbf{x}-\mathbf{x}_n)/\sigma\|^2\right)}{\sum_{n'=1}^{N}\exp\left(-\frac{1}{2}\|(\mathbf{x}-\mathbf{x}_{n'})/\sigma\|^2\right)}$ — *Post. prob. (E step)*
    $\mathbf{x} \leftarrow \sum_{n=1}^{N} p(n|\mathbf{x})\mathbf{x}_n$ — *Update* $\mathbf{x}$ *(M step)*
  until $\mathbf{x}$'s update $<$ `tol`
  $\mathbf{z}_n \leftarrow \mathbf{x}$ — *Mode*
end
connected-components($\{\mathbf{z}_n\}_{n=1}^N$,`min_diff`) — *Clusters*

3 Overview

GMS is slow: $\mathcal{O}(kN^2D)$. Computational bottlenecks:

B1: large average number of iterations $k \sim 100$ (linear convergence).

B2: large cost per iteration $\sim 2ND$ multiplications (E step: ND to obtain $p(n|\mathbf{x})$, M step: ND to obtain $\mathbf{x}^{(\tau+1)}$).

Acceleration techniques must address **B1** and/or **B2**.

Evaluation of strategies:

- Goal: to achieve the same segmentation as GMS. Visual evaluation of segmentation not enough; we compute the segmentation error wrt GMS segmentation (= no. pixels misclustered as a % over the whole image).
- We report running times in normalised iterations (= 1 iteration of GMS) to ensure independence from implementation details.
- No pre- or postprocessing of clusters (e.g. removal of small clusters).

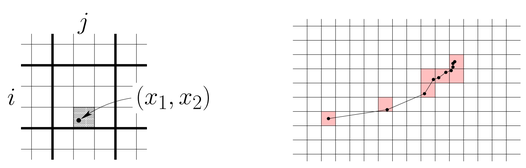
Why segmentation errors?

- The accelerated iterative scheme may not converge to a mode of $p(\mathbf{x})$.
- A point $\mathbf{x}_n$ may converge to a different mode than with exact GMS.

Strategy	Cost per iteration relative to exact GMS
`ms1`: spatial discretisation	1
`ms2`: spatial neighbourhood	e
`ms3`: sparse EM	2 if full step, e if partial step
`ms4`: EM–Newton	1 if EM step, $\left(1+\frac{D+1}{4}\right)$ if Newton step, $\left(\frac{3}{2}+\frac{D+1}{4}\right)$ if EM step after failed Newton step

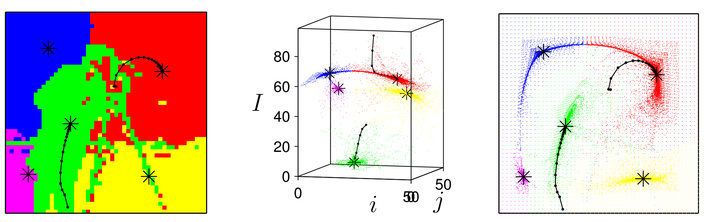
$e \in (0,1]$ is the fraction of the data set used (neighbours for `ms2`, plausible set for `ms3`).

4 `ms1`: spatial discretisation

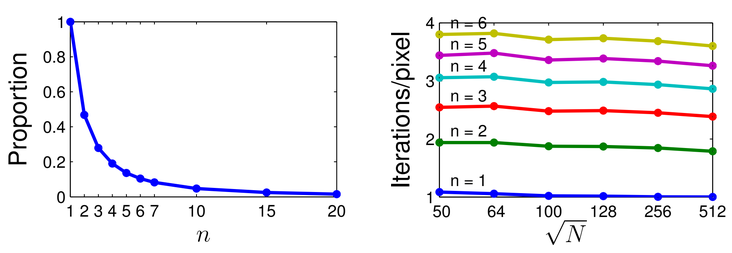


Idea: many different pixels converging to the same mode travel almost identical paths. We discretise the spatial domain by subdividing every pixel (i,j) into $n \times n$ cells; points projecting to the same cell share the same fate. This works because paths in D dimensions are well approximated by their 2D projection on the spatial domain. It massively reduces the total number of iterations because we stop iterating once we hit an already visited cell. We start first with a set pixels uniformly distributed over the image (this finds all modes quickly).

- Converges to a mode
- Segmentation error $\to 0$ by increasing n
- Addresses **B1**. Parameter: $n = 1, 2, 3\dots$ (discretisation level)



Plot of all the iterates for all starting pixels in (i,j,I) space ($*$ = modes). Most iterates concentrate on small regions, thus most cells in `ms1` are empty (1 pixel = $n \times n$ cells).



Left: proportion of cells (out of n^2N cells) visited by `ms1`, as a function of n: less than nN cells are visited. *Right*: average number of iterations per pixel as a function of the image size N.

5 `ms2`: spatial neigh.

Approximates E and M steps with a subset of the data points (rather than all N) consisting of a neighbourhood in the spatial domain (not the range domain). Finding neighbours is for free (unlike finding neighbours in full space).

- Does not converge to a mode
- Segmentation error $\to 0$ by increasing e
- Addresses **B2**. Parameter: $e \in (0,1]$ (fraction of data set used as neighbours)

6 `ms3`: sparse EM

Sparse EM (Neal & Hinton '98): coordinate ascent on the space of $(\mathbf{x}, \tilde{\mathbf{p}})$ where $\tilde{\mathbf{p}}$ are posterior probabilities; this maximises the free energy $F(\tilde{\mathbf{p}}, \mathbf{x}) = \log p(\mathbf{x}) - \mathrm{D}\left(\tilde{\mathbf{p}} \| p(n|\mathbf{x})\right)$ and also $p(\mathbf{x})$.

- Run partial E steps frequently, where we update $p(n|\mathbf{x})$ only for $n \in S$; S is the plausible set (nearest neighbours), kept constant over partial E steps. FAST.
- Run full E steps infrequently, which update all $p(n|\mathbf{x})$ and also S. SLOW.

We choose S containing as many neighbours as necessary to account for a total probability $1-\epsilon \in (0,1]$. Thus, the fraction of data used e varies after each full step.

- Converges to a mode no matter how S is chosen; computational savings if few full steps
- Segmentation error $\to 0$ by decreasing ϵ
- Addresses **B2**. Parameter: $\epsilon \in [0,1)$ (prob. not in S)

7 `ms4`: EM–Newton

Start with EM steps, which quickly increase p. Switch to Newton steps when EM slows down (reverting to EM if bad Newton step). Specifically, try Newton step when $\|\mathbf{x}^{(\tau)} - \mathbf{x}^{(\tau-1)}\| < \theta$. The Hessian of p has a simple form. Computing the Newton step yields the EM step for free.

- Converges to a mode with quadratic rate
- Segmentation error $\to 0$ by decreasing θ
- Addresses **B1**. Parameter: $\theta > 0$ (minimum EM step length) relative to σ

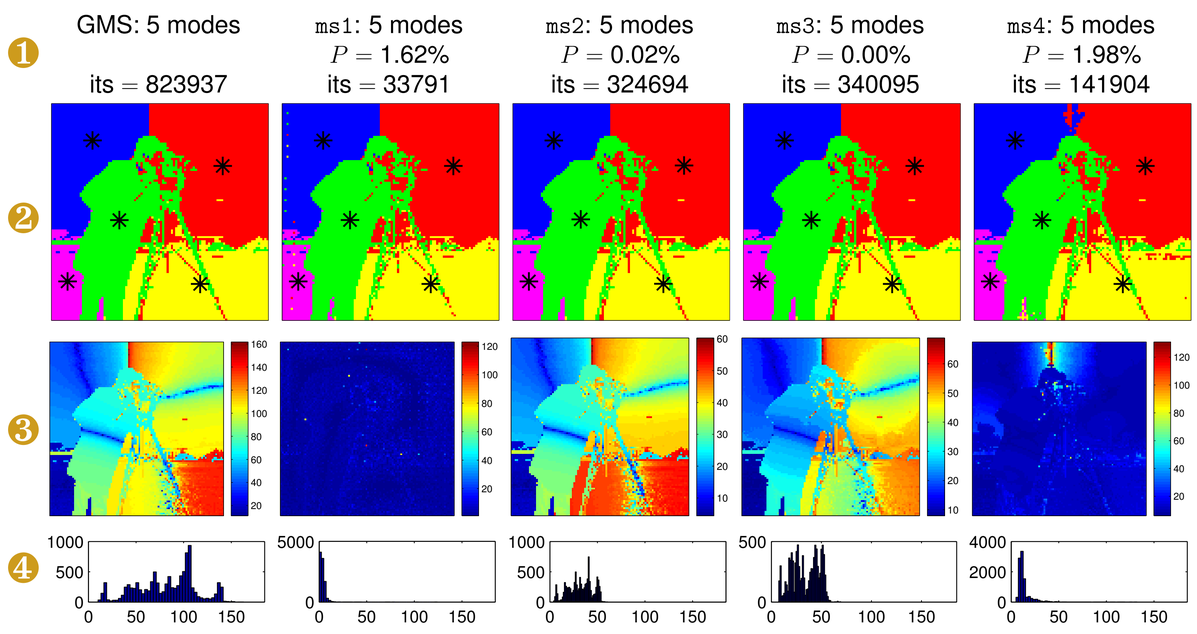
8 Experimental results

Dataset: $\mathbf{x}_n = (i_n, j_n, I_n)$ (greyscale) or $\mathbf{x}_n = (i_n, j_n, L_n^*, u_n^*, v_n^*)$ (colour) where (i,j) is the pixel's spatial position. Best segmentations appear for large bandwidths: $\sigma \approx \frac{1}{5}\times$(image side). We study all strategies with different images over a range of σ. Overall results:

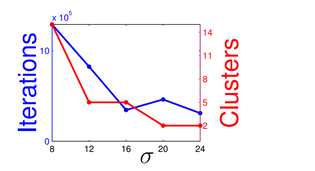
- With parameters set optimally, all methods can obtain nearly the same segmentation as GMS with significant speedups; near-optimal parameter values can easily be set in advance for `ms1`, `ms3` and `ms4`, and somewhat more heuristically for `ms2`.
- `ms1` attains the largest speedup by far: 10×–100× (average number of iterations per pixel k = 2–4 only) with segmentation error $<$ 3%. `ms4` ranks second best (1.5×–6× speedups). The neighbourhood methods (`ms2`, `ms3`) attain more modest speedups (1×–3×); of these, `ms3` attains the lowest (near-zero) error of all strategies, while `ms2` can result in unacceptably large errors for a suboptimal setting of its parameter. For the larger bandwidths, the neighbourhood methods do not improve over GMS, since the neighbourhood size e becomes comparable to the data set size.

Other methods:

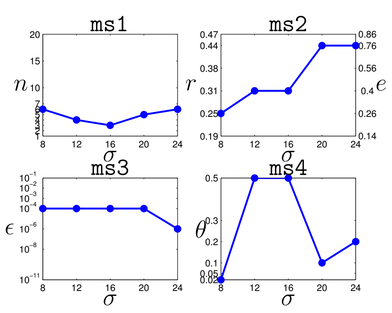
- Approximate algorithms for neighbour search (e.g. kd-trees): less effective because large neighbourhood.
- Other optimisation algorithms (e.g. quasi-Newton): need to ensure first steps are EM.
- Fast Gauss transform: can be combined with our methods.



Segmentation results for each method under its optimal parameter value for $\sigma = 12$. For each method we give: ❶ the number of modes, error P and number of normalised iterations; ❷ the colour-coded segmentation with modes marked $*$; and the distribution of the number of normalised iterations at each pixel, ❸ over the image and ❹ as a histogram.



Total number of iterations and number of clusters for GMS as a function of the bandwidth σ.



Optimal parameter value for each method (= that which minimises the iterations ratio subject to achieving an error $P < 3\%$) as a function of σ.

Clustering error P (percent) and computational cost for each method as a function of its parameter.

Clustering error P (percent) and number of iterations for each method under its optimal parameter value as a function of σ.

9 Conclusion

The best method is `ms1` spatial discretisation (possibly combined with `ms4` EM–Newton), which can accelerate GMS by one to two orders of magnitude while achieving essentially the same segmentation. Neighbourhood methods (`ms2` spatial neighbourhood, `ms3` sparse EM) are less effective because GMS needs very large neighbourhoods. All 4 methods are readily extended to adaptive and non-isotropic bandwidths. `ms2` and `ms3` are also applicable to clustering non-image data.

Partially supported by NSF CAREER award IIS–0546857.