ACCV2002: The 5th Asian Conference on Computer Vision, 23–25 January 2002, Melbourne, Australia

Algorithm Evaluation by Probabilistic Fitness/Cost Analysis, and Application to Image Segmentation

Mark Everingham, Henk Muller, and Barry Thomas *
Advanced Computing Research Centre
University of Bristol
Bristol, BS8 1UB, UK.
everingm@cs.bris.ac.uk

Abstract

In many areas of computer vision the research producing new algorithms greatly exceeds work on their evaluation. Evaluation of vision algorithms is often difficult because of the multiple objectives that an algorithm should meet, for example accuracy and computational efficiency, and because algorithms typically have several parameters which must be specified by the user. In this paper we propose a framework for evaluation of algorithms with multiple objectives, which allows probabilistic analysis of the behavior of a set of algorithms in a joint fitness/cost space. We take the image segmentation problem as an example application domain and use our approach to compare seven state-of-the-art image segmentation algorithms.

1. Introduction

Many areas of computer vision demand algorithms which have multiple objectives, for example producing an output of satisfactory accuracy within a reasonable amount of processing time. While new algorithms are published in great numbers, little attention has been directed at their evaluation. Evaluation of algorithms having multiple objectives can be difficult because of uncertainty in what trade-offs can satisfactorily be made between the objectives. The difficulty is compounded if there are several algorithm parameters which must be specified by the user.

In this paper we propose a new framework which can be used to evaluate algorithms probabilistically in a multi-dimensional fitness/cost space without having to define exact weights between the multiple objectives. The novel aspect of this framework is that we use a probabilistic approach to allow examination of an algorithm's stability in addition to its average performance. This gives a significant improvement over the evaluation by "monotonic hulls" which we have proposed [3], and is more general than the convex hull methods which have been proposed for analysis of receiver operating characteristic (ROC) curves [8]. We apply our approach to the evaluation of image segmentation algorithms, since this is an area in which some attempts at evaluation have been made, and show that our approach offers more informative results in comparison with other published methods.

*Part of this work was sponsored by Hewlett-Packard Research Laboratories, Bristol

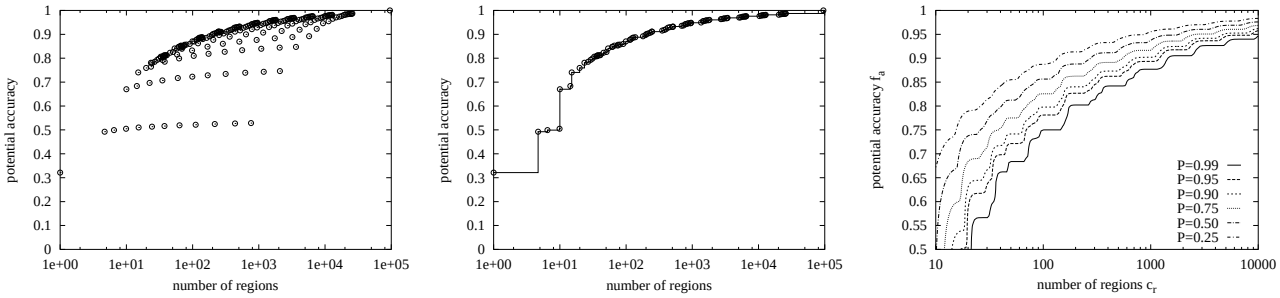
2. Evaluating segmentation algorithms

Image segmentation is the first stage of processing in many practical computer vision systems. Over the last few decades many segmentation algorithms have been developed, with the number growing steadily every year. Typically the effectiveness of a new algorithm is demonstrated only by the presentation of a few segmented images, allowing only subjective and qualitative conclusions to be drawn about that algorithm. We believe that if segmentation algorithms are to be successfully applied in real vision systems, quantitative assessment methods of algorithms need to be defined.

Zhang [10] has proposed a classification of existing quantitative evaluation methods:

"Analytical" methods attempt to characterize an algorithm in terms of principles, requirements, complexity, etc. without reference to a concrete implementation of the algorithm, or test data. For example, one can define the time complexity of an algorithm or its response to a theoretical data model such as a step edge. While in domains such as edge detection this may be useful, in general the lack of a general theory of image segmentation limits these methods.

"Empirical goodness" methods evaluate algorithms by



a) Fitness/cost b) Monotonic hull c) Cumulative probability distribution

Figure 1. Example fitness/cost graph, monotonic hull, and cumulative probability distribution

computing a "goodness" metric on the segmented image, for example grey-level uniformity of the output regions [6], without *a priori* knowledge of the desired segmentation result. The advantage of this class of methods is that they require only that the user defines a goodness metric, and can be used for online evaluation. The great disadvantage is that the goodness metrics are at best heuristics, since no ground truth data is available, and may exhibit strong bias towards a particular algorithm.

"Empirical discrepancy" methods calculate some measure of discrepancy between the segmented image output by an algorithm and the correct segmentation desired for the corresponding input image. The use of ground truth potentially makes this class of methods the most general and least biased. Proposed discrepancy measures include pixel class confusion [9], distance to correctly segmented pixels [9], or differences in feature values measured from regions [11]. The principle drawback with these methods is that one must manually define the ground truth segmentation, unless working with synthetic image data.

2.1. Limitations

We believe that the main problem with previous approaches is their attempt to combine the multiple objectives of an algorithm into a single metric, where in reality we expect that we always have to make a trade-off between different properties of the algorithm. Methods which are based solely on a quality metric do not allow such trade-offs to be evaluated objectively.

Our approach, described in the next section, does not fit easily into any one of Zhang's categories [10] but can be seen as a unification of all categories into a consistent framework, defining a general methodology of comparison incorporating multiple measures rather than advocating the use of a single particular measure. Our approach is most similar to the "empirical discrepancy" methods, but importantly has the distinction of not defining just a single discrepancy metric and evaluating effectiveness in a discrepancy/parameter space, but instead performing a probabilistic evaluation in a multi-dimensional fitness/cost space.

3. Algorithm evaluation in fitness/cost space

Rather than attempt to define very specific measures of an algorithm's performance, we start with a very general form of overall fitness

$$H(a_{\vec{p}}, D) = \Phi\left(f_1(a_{\vec{p}}, D) \dots f_m(a_{\vec{p}}, D), c_1(a_{\vec{p}}, D) \dots c_n(a_{\vec{p}}, D)\right) \quad (1)$$

where $a_{\vec{p}}$ is an algorithm a instantiated by parameters $\vec{p}$ and D is a set of test data for which the desired output is known. Functions $f_i(a_{\vec{p}}, D)$ are individual fitness functions from a set $F = \{f_1, \dots, f_m\}$ and are defined to increase monotonically with the *fitness* of some particular aspect of the algorithm's behavior. Functions $c_i(a_{\vec{p}}, D)$ are individual cost functions from a set $C = \{c_1, \dots, c_n\}$ and are defined to increase monotonically with the *cost* of some particular aspect of the algorithm's behavior. Cost functions c_i could equivalently be defined as negative fitness functions, but we make the distinction here for the sake of clarity.

Φ combines the individual fitness and cost functions into an overall measure of fitness. We believe that in general it is difficult to specify an exact form for Φ, since this requires defining the exact trade-off to be made between fitness and costs, and therefore assume it to be of unknown form. With no loss of generality we assume that Φ increases monotonically with increasing values of all fitness functions $f \in F$ and decreases monotonically with increasing values of all cost functions $c \in C$.

We can plot a projection of Φ onto a single fitness function f_i and cost function c_j for different parameters $\vec{p}$ by a zero-scaling of all other functions. Figure 1a shows an example where a potential accuracy function f is plotted against the mean number of regions c output per image for an image segmentation algorithm (see Section 6.1), by sampling the parameter space of the algorithm.

3.1. Monotonic hull

Many of the points on the graph of Figure 1a are intuitively undesirable since they have lower fitness and higher cost than points corresponding to other choices of parameters. In the 2-D case these undesirable parameter settings

are points which are to the bottom-right of another point. Since we have defined our overall fitness function Φ to be monotonic, then for a particular algorithm a with parameters taken from the set P_a it is readily shown that the only worthwhile choices of parameters are in the set

$$\{\vec{p} \in P_a | \neg\exists \vec{q} \in P_a : n(a_{\vec{q}}, a_{\vec{p}})\} \tag{2}$$

where

$$n(a_{\vec{q}}, a_{\vec{p}}) = \left\{ \begin{array}{ll} \text{true} & \text{if } \left\{ \begin{array}{l} \forall f \in F : f(a_{\vec{q}}, D) \geq f(a_{\vec{p}}, D) \wedge \\ \forall c \in C : c(a_{\vec{q}}, D) \leq c(a_{\vec{p}}, D) \end{array} \right\} \\ \text{false} & \text{otherwise} \end{array} \right. \tag{3}$$

and $F = \{f_1 \dots f_m\}$, $C = \{c_1 \dots c_m\}$. We call the set of such points the "monotonic hull" [3]. Figure 1b shows these points for the graph of Figure 1a, where we have drawn connecting lines to indicate the partitioning of the space by the hull.

The strength of this construction is that we can readily show that for *any* choice of overall fitness function Φ that increases monotonically in F and decreases monotonically in C, parameters which do not correspond to points on the hull are *always* bad choices, since there is another point on the hull which increases the overall fitness.

We can extend Equation 2 to the case of multiple algorithms, and define the monotonic hull over both algorithms a and their parameters $\vec{p}$:

$$\{a_{\vec{p}} | a \in A \wedge \vec{p} \in P_a \wedge \neg\exists b \in A, \vec{q} \in P_b : n(b_{\vec{q}}, a_{\vec{p}})\} \tag{4}$$

This has the natural consequence that any *algorithm* that does not fall on the monotonic hull is similarly a bad choice for *any* monotonic fitness function Φ.

4. Probabilistic evaluation

In Section 3.1 we assumed that the test data D is atomic such that a particular instantiation of an algorithm $a_{\vec{p}}$ results in a single point in fitness/cost space. In many applications however, there is a natural grouping of the test data D into subsets of data $d \in D$, for example if D is a set of images and $a_{\vec{p}}$ is an instantiation of an image segmentation algorithm, then it is desirable not only to look at the *average* performance of the algorithm over all images, but also how *stable* its performance is over a set of images, or equivalently how much the performance varies over individual images $d \in D$. If only the average fitness/cost results are used, then an algorithm which performs with great stability (low variability) but slightly lower average performance than one which has lower stability (greater variability) but slightly higher average performance may be disregarded entirely.

4.1. Probability distribution in fitness/cost space

In fitness/cost space we are not concerned with the probability that an algorithm yields a particular point in that space, but rather with the probability that it gives costs no higher than some set of thresholds, and fitnesses no lower than another set of thresholds. If we assume that for a particular instantiation of an algorithm by its parameters $a_{\vec{p}}$ the fitnesses and costs are mutually independent, then we obtain

$$P\left(\forall c \in C : c(a_{\vec{p}}, S) \leq k_c \wedge \forall f \in F : f(a_{\vec{p}}, S) > k_f\right) = \prod_{c \in C} P\left(c(a_{\vec{p}}, S) \leq k_c\right) \prod_{f \in F} P\left(f(a_{\vec{p}}, S) > k_f\right) \tag{5}$$

Note that while in general the independence assumption may not hold, since it is only made locally around a single instantiation of an algorithm by its parameters, we believe it to be a satisfactory approximation.

Taking the maximum of Equation 5 over all parameters $\vec{p} \in P_a$ gives an expression for the overall behavior of an algorithm a if we are allowed to set the parameters $\vec{p}$ to optimum values:

$$P\left(\forall c \in C : c(a, S) \leq k_c \wedge \forall f \in F : f(a, S) > k_f\right) = \max_{\vec{p} \in P_a} P\left(\forall c \in C : c(a_{\vec{p}}, S) \leq k_c \wedge \forall f \in F : f(a_{\vec{p}}, S) > k_f\right) \tag{6}$$

This expresses with what probability an algorithm can achieve an output with at worst specified fitness/costs given a suitable choice of parameters.

If we further extend Equation 6 to cover a *set* of algorithms A, we obtain:

$$P\left(\forall c \in C : c(A, S) \leq k_c \wedge \forall f \in F : f(A, S) > k_f\right) = \max_{a \in A, \vec{p} \in P_a} P\left(\forall c \in C : c(a_{\vec{p}}, S) \leq k_c \wedge \forall f \in F : f(a_{\vec{p}}, S) > k_f\right) \tag{7}$$

This expresses with what probability some algorithm in the set A can achieve an output with at worst specified fitness/costs.

4.2. Fitness/cost probability distributions

We consider two specific forms for modelling the individual cost/fitness probability distributions $p(f = t | a_{\vec{p}})$. In the case that the fitness/cost f for an instantiation of an algorithm $a_{\vec{p}}$ is not guaranteed to be constant across inputs $d \in D$, we assume a Gaussian distribution:

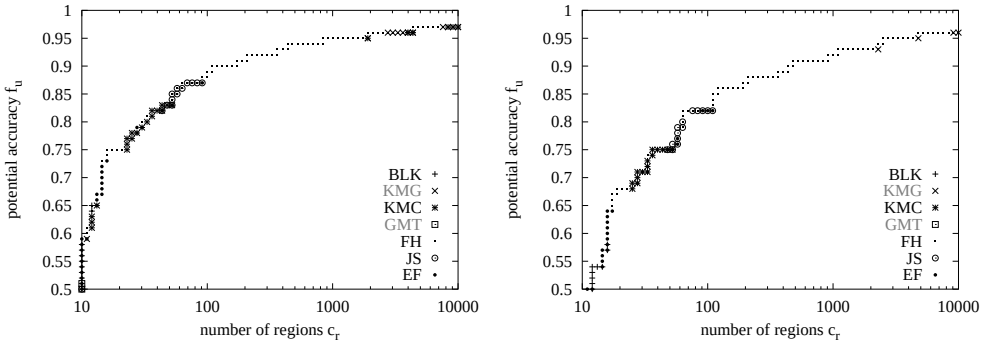
$$p(f = t | a_{\vec{p}}) = G(t, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(t-\mu)^2}{2\sigma^2}\right\} \tag{8}$$

where we estimate the mean μ and variance σ^2 from the test data D. The cumulative distribution for a fitness function f is then given by

$$P\left(f(a_{\vec{p}}, D) > k\right) = 1 - \int_{-\infty}^{k} G\left(t, \mu(f, a_{\vec{p}}, D), \sigma^2(f, a_{\vec{p}}, D)\right) \, \mathrm{d}t \tag{9}$$

and equivalently for a cost function c:

$$P\left(c(a_{\vec{p}}, D) <= k\right) = \int_{-\infty}^{k} G\left(t, \mu(c, a_{\vec{p}}, D), \sigma^2(c, a_{\vec{p}}, D)\right) \, \mathrm{d}t \tag{10}$$



a) $P = 0.5$ b) $P = 0.9$

Figure 2. Probabilistic hulls $f_u \times c_r$ at $P = 0.5$, $P = 0.9$

Certain algorithms produce an output for which some fitness/cost functions are constant for a particular choice of parameters $\vec{p}$ (for example, algorithm BLK in Section 6.2). In this case, the probability distribution $p(f = t|a_{\vec{p}})$ has the form of the Dirac delta function

$$p(f = t|a_{\vec{p}}) = \delta\left(t - \hat{f}(f, a_{\vec{p}}, S)\right) \tag{11}$$

where

$$\delta(t) = 0 \text{ if } t \neq 0; \int_{-\infty}^{\infty} \delta(t)\,\mathrm{d}t = 1 \tag{12}$$

and $\hat{f}$ is the constant fitness/cost value for the instantiation of the algorithm $a_{\vec{p}}$ run on test data D. The cumulative distribution for a fitness function f is then given by the Heaviside step function:

$$P\left(f(a_{\vec{p}}, S) > k\right) = 1 - H\left(k - \hat{f}(f, a_{\vec{p}}, S)\right) \tag{13}$$

where we define

$$H(t) = \begin{cases} 0 & : \ t < 0 \\ 1 & : \ t \geq 0 \end{cases} \tag{14}$$

and equivalently for a cost function c:

$$P\left(c(a_{\vec{p}}, S) \leq k\right) = H\left(k - \hat{c}(c, a_{\vec{p}}, S)\right) \tag{15}$$

5. Uses of probability distribution

5.1. Coverage of fitness/cost space

Evaluation of the probability distribution for a single algorithm (Equation 6) gives us the probability with which that algorithm can produce an output with at worst specified fitness/costs. Figure 1c shows an example in a 2-D fitness/cost space (Section 6.1), with contours at the indicated probability levels. In this case we see that the fitness has high variance for low cost, decreasing as the cost increases. The desirable points of low cost and high fitness (top left of graph) are covered only with low probability ($P < 0.25$)

If we use the distribution over a set of algorithms (Equation 7), we can see with what probability any part of the fitness/cost space is covered by an algorithm from the set.

5.2. Selection of algorithm and parameters

If we fix a set of minimum fitness thresholds $\{k_f\}$ and maximum cost thresholds $\{k_c\}$ then the algorithm and parameter choice $\hat{a_{\vec{p}}} \in \{A, P_a\}$ which maximizes the probability of achieving these objectives can be chosen by Equation 5:

$$\begin{array}{l} \hat{a_{\vec{p}}} = \arg\max_{a \in A, p \in P_a} \\ P\left(\forall c \in C : c(a_{\vec{p}}, S) \leq k_c \wedge \forall f \in F : f(a_{\vec{p}}, S) > k_f\right) \end{array} \tag{16}$$

By determining $\hat{a_{\vec{p}}}$ for any point in the fitness/cost space we can determine which algorithm performs with most stability in an area of the space. An example can be seen in Figure 4 (Section 6.3).

5.3. Probabilistic hull

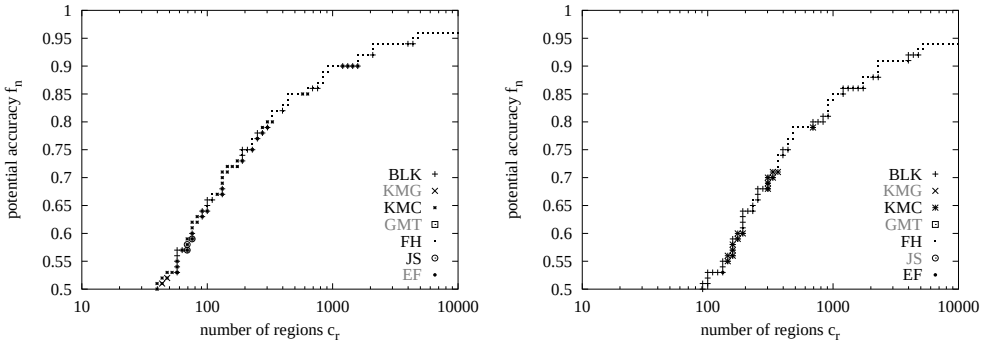
By placing a threshold t_P on the probability distribution across a single algorithm (Equation 6), or set of algorithms (Equation 7), and evaluating the monotonic hull of those points in fitness/cost space at which the probability is above threshold, we obtain a hull labelled by the algorithms having above threshold probability at that point. We call this a "probabilistic hull".

The probabilistic hull has an equivalent interpretation to that of the monotonic hull (Section 3.1). Any instantiation of an algorithm $a_{\vec{p}}$ which does not contribute to the hull is *always* a bad choice under any monotonic fitness function Φ (Equation 1), since another algorithm $a_{\vec{q}}$ yields a better output with at least the specified probability t_P. An example is shown in Figure 2 (Section 6.3).

6. Experiments

6.1. Fitness and cost functions

In our experiments we examine image segmentation algorithms, and define the objective of the algorithms to be to form regions such that if the optimal object label is assigned to all pixels of a region, as much as possible of the



a) $P = 0.5$ b) $P = 0.9$

Figure 3. Probabilistic hulls $f_n \times c_r$ at $P = 0.5$, $P = 0.9$

object set is correctly labelled. The object label is a unique identifier of each object present in the images. The test data is a set of color images of outdoor urban scenes, and their manual segmentation into objects according to a set of eight classes including road, pavement, vehicle, etc. Our fitness function expressing *potential* accuracy for a single image is the following:

$$f(a_{\vec{P}}, i) = \frac{\sum_{o \in O_i} \frac{|\{x \in o | \hat{o}_x = o\}|}{|o|} |w(o)}{\sum_{o \in O_i} w(o)} \quad (17)$$

where O_i is the set of objects in image i, an object $o \in O_i$ is a set of pixels, and $\hat{o}_x$ is the optimal object label of a pixel x which maximizes the overall measure. The contribution of each object to the measure is weighted by a function w for which we have tried several forms. Setting $w(o) = |o|$ gives the proportion of image pixels that could be correctly labelled. This is the same as has been used in other methods [9], but may introduce bias in that objects or classes of object having many pixels contribute more to the measure than those with fewer pixels. We can remove this bias by defining w thus:

$$w(o) = \begin{cases} \frac{1}{P(\lambda_o)|o|} & \text{if } |o| \geq k \\ 0 & \text{if } |o| < k \end{cases} \quad (18)$$

where λ_o is the class label of an object and $P(\lambda)$ the prior probability of an object being drawn from class λ. The constant k causes objects with smaller pixel area than k to be discarded, which prevents very small objects biasing the metric. We must define k manually, but it can remain constant across different images and segmentation algorithms. In this form we measure the *proportion* of each object which can potentially be correctly labelled. We term un-normalized fitness f_u, and normalized by Equation 18 f_n.

For the sake of brevity we report results on two simple cost functions here: c_r =number of regions, and c_t =segmentation time. The number of regions characterizes over-segmentation and the required post-processing time for a classification system, assuming constant time per output region.

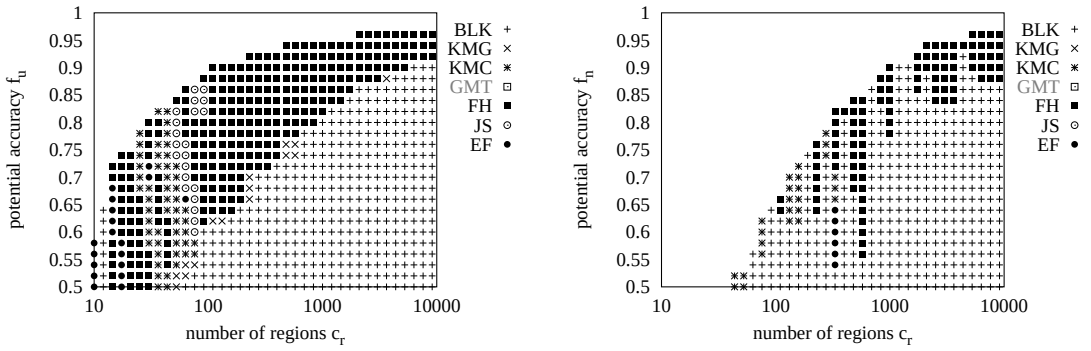
6.2. Segmentation algorithms

We ran experiments with seven segmentation algorithms: three state-of-the-art algorithms for which implementations are publicly available, and four simple or fast algorithms. BLK simply divides an image statically into square blocks of constant size. It has zero run-time since the segmentation is independent of the input image, and is a useful baseline. KMG, KMC, and GMT cluster low-level features, form connected regions, and merge regions to a minimum region size. KMG uses grey-level features alone and the K-means clustering algorithm [1]. KMC uses color features and K-means. GMT uses color and texture features from a bank of Gabor filters [5], and a Gaussian mixture model [1] for clustering. FH [4] uses dynamic programming to form regions which are guaranteed to be neither too coarse nor fine with respect to a color edge strength measure, then merges regions to a minimum region size. JS [2] uses color quantization followed by a multi-scale region growing step which aims to segment both uniform and textured regions. EF [7] uses predictive coding to estimate the direction of change in color and Gabor texture features, and forms boundaries by propagating the flow field.

6.3. Results

Figure 2 shows the probabilistic hulls for all algorithms in the $f_u \times c_r$ space at minimum probabilities $P = 0.5$ (Figure 2a) and $P = 0.9$ (Figure 2b). We have limited the y-axis to minimum 50% potential accuracy and the number of regions to maximum 10,000 for the sake of clarity, since we argue that a segmentation result outside these ranges is unlikely to be useful. Algorithms which do not contribute to the hull are shown grayed in the key.

We can see that to potentially label 80% of the image correctly with probability greater than 0.5, we need to seg-



a) $f_u \times c_r$ b) $f_n \times c_r$

Figure 4. Algorithm stability in fitness/cost space

ment into around 20 regions, and to reach 90% around 100 regions are needed. Figure 2b clearly shows the value of using the probabilistic analysis: If we require results with probability greater then 0.9, then to potentially label 80% of the image correctly we need to segment into around 60 regions, and to reach 90% around 400 regions are needed, a factor of four greater than in the mean case.

FH dominates the hull, contributing points across the whole range of cost. Algorithms KMC, EF and JS contribute small sections to the hull, out-performing FH for lower numbers of regions. Each appears to have a defined window in terms of the number of regions over which it performs well, while for higher numbers of regions FH gives consistently higher potential accuracy. In the case of the $P = 0.9$ hull, we see that BLK forms part of the hull for very small numbers of regions. This is unsurprising since this algorithm guarantees the number of regions output, but the accuracy at this level is low ($< 55\%$). Two algorithms are absent from the hull: KMG and GMT, which use grey-level and color/texture respectively. We found KMG had a tendency to over-segment, perhaps because of the lack of color information, and GMT had high variance in fitness/cost space, yielding uncertain results.

Figure 3 shows the equivalent hulls at $P = 0.5$ (Figure 3a) and $P = 0.9$ (Figure 3b) for the normalized potential accuracy metric. Of note is that the number of regions required to achieve a given level of potential accuracy is higher than in the un-normalized metric, by a factor of 2–3. Only three algorithms contribute significantly to the hulls (JS and EF are on the hull in a tiny area): FH, KMC, and BLK. At $P = 0.5$ FH dominates for higher accuracy, and KMC for lower number of regions, while at $P = 0.9$ FH again dominates the hull since its output is more stable than that of KMC. Interestingly the BLK algorithm also appears on the hulls across a wide range of accuracy, particular at $P = 0.9$. Despite its naive method, BLK does have an advantage of providing a guaranteed number of regions. It may also perform better according to the normalized metric since it cannot have any bias towards forming larger regions, bias which the other algorithms can be seen to exhibit.

Figure 4 shows which algorithms perform with most stability at points in the fitness/cost space, according to Equation 16. Parts of the space for which no algorithm produces an output with $P > 0.5$ are left blank. In the case of the un-normalized f_u accuracy metric (Figure 4a), FH produces the most stable results over most of the space, with KMC and JS more stable for a narrow band in the lower numbers of regions, and EF more stable in a small area of the space. The difference in probability level between FH and the other algorithms in this area also proved small, suggesting that FH is a good choice regardless of the fitness/cost trade-off made here. In the case of the normalized accuracy metric, the picture is less clear, with FH, KMC and BLK performing with most stability in bands according to the number of regions. Differences between the probability levels of the algorithms were again small, but the plot suggests no algorithm is performing consistently with respect to this metric.

We conclude with an evaluation of the algorithms in the 3-D $f_u \times c_r \times c_t$ space. This allows evaluation of the algorithms' potential accuracy, output complexity, and processing time. Figure 5 shows a view of the probabilistic hull for $P = 0.5$. Results for the normalized accuracy metric f_n and higher probability threshold show the same pattern, and are omitted here for the sake of brevity.

In this case, where processing time becomes a factor in an algorithm's evaluation, we can see that this factor is a central difference between algorithms. FH gives highest potential accuracy for low numbers of regions, but has high processing time. KMC offers comparable performance, but is an order of magnitude faster, and KMG is another order of magnitude faster than KMC for a slight reduction in potential accuracy. BLK, which has zero processing time since the segmentation does not vary with the input image, dominates the hull if very low processing time is required,

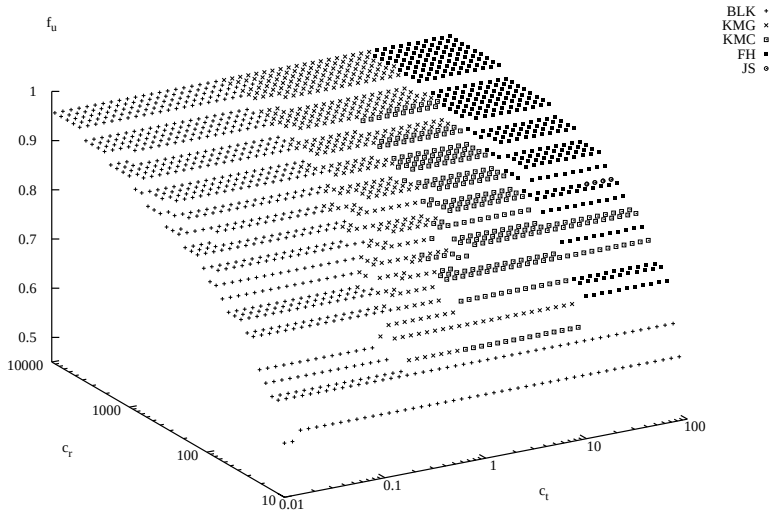


Figure 5. Probabilistic hull $f_u \times c_r \times c_t$ at $P = 0.5$

and can surprisingly offer comparable accuracy to the other algorithms for moderate numbers of regions.

7. Conclusions

We have proposed methods for evaluating algorithms having multiple objectives by probabilistic analysis in a multi-dimensional fitness/cost space. The key advantage over previous methods [3, 8] is that we can evaluate the stability of algorithms in addition to their average performance. This is crucial for real world applications in which we need to be more sure about the performance we can expect from an algorithm than we can ascertain by an average-case analysis.

We applied our approach to the evaluation of image segmentation algorithms, and showed that more informative results could be obtained using the new approach in comparison with analysis using the monotonic hull [3]. In these experiments we found that one algorithm, FH [4], outperformed the other algorithms tested overall, producing more accurate and stable output than the other algorithms. This is an appealing result because of the algorithm's strong theoretical basis [4]. In addition, we showed that in certain areas of the fitness/cost space, several simpler algorithms with lower computational cost produce more stable results than FH. Results of this kind, obtained by the use of probabilistic analysis, allow more informed choices of algorithm and parameters to be made for a particular application.

Further work in the domain of image segmentation evaluation certainly remains to be done on refining the fitness/cost metrics used. We believe that, in conjunction with such work, the probabilistic hull methods we have described offer useful, more informative, and more general tools for algorithm evaluation than previously proposed. We intend to further investigate this domain, and the application of the approach to other multiple-objective vision problems.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[2] Y. Deng, B. S. Manjunath, and H. Shin. Color image segmentation. In *Proc. CVPR'99*, pp. 446–451, 1999.

[3] M. R. Everingham, H. Muller, and B. T. Thomas. Evaluating image segmentation algorithms using monotonic hulls in fitness/cost space. In *Proc. BMVC'2001*, pp. 363–372, 2001.

[4] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *Proc. CVPR'98*, pp. 98–104, 1998.

[5] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[6] M. D. Levine and A. Nazif. Dynamic measurement of computer generated image segmentations. *IEEE Trans. PAMI*, 7:155–164, 1985.

[7] W. Y. Ma and B. S. Manjunath. EdgeFlow: A technique for boundary detection and segmentation. *IEEE Trans. IP*, 9(8):1375–1388, 2000.

[8] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proc. KDD'97*, 1997.

[9] W. A. Yasnoff, J. K. Mui, and J. W. Bacus. Error measures for scene segmentation. *Pattern Recognition*, 9(4):217–231, 1977.

[10] Y. J. Zhang. A survey on evaluation mehods for image segmentation. *Pattern Recognition*, 29(8):1335–1346, 1996.

[11] Y. J. Zhang and J. J. Gerbrands. Objective and quantitative segmentation evaluation and comparison. *Signal Processing*, 39(1–2):43–54, 1994.