IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No. 10, pp. 1075–1088, October 2001.

GLOBALLY OPTIMAL REGIONS AND BOUNDARIES AS MINIMUM RATIO WEIGHT CYCLES

IAN H. JERMYN AND HIROSHI ISHIKAWA

Abstract. We describe a new form of energy functional for the modelling and identification of regions in images. The energy is defined on the space of boundaries in the image domain, and can incorporate very general combinations of modelling information both from the boundary (intensity gradients,...), *and* from the interior of the region (texture, homogeneity,...).

We describe two polynomial-time digraph algorithms for finding the *global* minima of this energy. One of the algorithms is completely general, minimizing the functional for any choice of modelling information. It runs in a few seconds on a 256×256 image. The other algorithm applies to a subclass of functionals, but has the advantage of being extremely parallelizable. Neither algorithm requires initialization.

1. Introduction

One of the fundamental problems in image understanding is to identify regions in images with particular semantic content. It is safe to say that if there was a mechanism to answer reliably the question "What are the regions in the image that correspond to instances of O?", where O is some named class of objects, then many other tasks in image understanding would be greatly simplified.

Approaches to this problem, which of course is very far from being solved, tend to break into two categories. One approach is to segment the image globally, partitioning the image domain into labelled subsets based on some, usually generic criteria, with the hope that subsequent processing can group and split these regions using more sophisticated models, and hence decompose the image domain into recognisable objects. These methods search a space of maps *from* the image domain to some other space.

Another approach to the same problem involves trying to identify objects in an image directly, by searching a space of structures mapped *into* the image. Here the emphasis is on modelling the properties of the regions occupied by objects from the outset, to various different levels of genericity. Examples of this approach are template-matching methods, and the many variations on active contours.

Although these approaches are not mathematically entirely distinct (a region can always be modelled by its characteristic function, which is a field on the image), they do correspond conceptually to distinctions in the human visual system. It seems likely for example [26, 29] that human perception of motion is based both on generic low-level computations, as well as on the identification and tracking of specific objects. They also lead to different visions of how to proceed: in the one case, the development of mechanisms to perform further organization of the generic segmentation; in the other, the development of more sophisticated and specific models for individual regions. The two approaches are in any case complementary, since generic segmentations can be used to inform the object models, as well as vice-versa.

Many of these methods are explicitly or implicitly framed as optimization problems. The difficulty is that these optimization problems cannot in general be solved globally, meaning that the solutions that are found have an unknown dependence on initial conditions. In the case of segmentation methods, this usually takes the form of a choice of a number of region seeds, whereas for active contours an initial contour is necessary. For contours, the only problem that has hitherto been solved globally for general energies is that of finding an optimal curve joining two given points. *A priori* such curves do not allow the identification of regions, requiring further processing to group them into boundaries.

Unfortunately, the problem of finding a globally optimal boundary in an image without loss of descriptive power in the model cannot trivially be solved by the application of the same kind of techniques that work in the open curve case. The topological constraint of closure is not so easily incorporated into these local algorithms. In addition, there are obstructions to solving the problem of globally minimizing the linear form of energy typically used for active contours, to wit, the elimination of self-intersections and repeated segments, and the existence of trivial solutions. (We discuss these issues more fully in section 4.1.) It is therefore of interest to have a model of region identification for which the global solution can be known, in order to form some judgement of the relevance of the solution for the image as a whole, independent of the initial conditions.

In addition, many of the above models utilise only one of two possible sources of information about the region: the properties of the interior or of the boundary. Many segmentation methods partition the image domain based on the similarity and dissimilarity of interior properties such as colour, intensity, or texture measures, while active contour methods typically utilise boundary properties such as intensity gradients. This distinction is by no means absolute, and many methods do introduce both types of information. This is often done on an ad hoc basis however, and the resulting optimization problems are almost never solved globally.

Motivated by the above considerations, and by psychological work, which since the Gestalt movement has emphasized the importance of contour closure in human vision [13, 14, 18, 6, 7], we propose an energy functional on boundaries in images that takes the form of a ratio of two integrals around the boundary. Our approach thus falls into the second category discussed above, that of region identification. (Curious readers may wish to look ahead to equation 1 at this point.) The numerator of the energy is a measure of the 'flow' of some quantity, for example intensity, into or out of the region, while the denominator is a generalised measure of the length of the boundary. This form turns out to have unexpected benefits. First, it solves the global optimisation problem: every instance of this form of energy can be globally optimised in polynomial time by the same graph algorithm, but the energy still allows for the incorporation of very general types of modelling information. In particular, because the energy is defined on closed curves and the numerator is a measure of flow, region information such as area, homogeneity and texture can be included in the energy as easily as can boundary information such as gradients, by the use of Green's theorem.

Two other properties of the energy are contingent on the exact nature of the region and boundary properties modelled. Simple but useful instances, and in particular the case where we incorporate information about boundary intensity gradients and boundary length only, are parameter-free, both at the level of the model and the algorithm. Despite being parameter-free, these instances include both data and regularisation terms; they eliminate a parameter by being a ratio of the two, rather than a linear combination as in standard active contour

energies. As will be seen in section 4, this behaviour is equivalent to automatic parameter selection for the linear active contour energy. More complex instances of the energy are of course parameterized.

A second contingent property, which we explore more fully in section 3.3, arises because the energy is a ratio of two integrals around the boundary: it can be 'scale-invariant'. By scale-invariant we mean that boundaries with different lengths but lying on the same data have the same energy, meaning that the energy does not have a bias towards long or short boundaries *a priori*. The simple instance involving intensity gradient and length information mentioned above possesses this property. This invariance can be broken, for example by the addition of an area term, but this becomes a modelling decision, rather than an uncontrolled feature of the energy itself as it is for linear energies.

In [12], a different algorithm was proposed for the optimization of this form of energy. This 'minimum mean weight cycle' algorithm [15] was not general, optimizing only a small subset of the possible instances of the energy, and in addition, had no sensible continuum interpretation. This introduced an undesirable dependence on the discretization, which manifested itself in unfortunate behaviour with respect to boundary length. In this paper, we introduce a different polynomial-time algorithm, the 'minimum ratio weight cycle' algorithm, that globally optimizes *any* instance of the energy. The new algorithm [19, 22] outperforms the old one not only in its much broader range of applicability, but even in its time and space requirements in the instances that the old algorithm could solve. We describe the previous algorithm here also, both for completeness and because it is extremely parallelizable, raising the prospect of a simple hardware implementation. Neither of these algorithms requires initialization. They find the global minimum of the discrete version of the energy functional, so that to within the accuracies allowed by the discretization they minimize the functional itself. No choice of seeds or initial contour is used.

The paper is laid out as follows. In the next section, we discuss related work. In section 3, we describe the form of energy functional, and some of its properties. We give some examples of the types of information that it can incorporate. In section 4 we discuss the graph algorithms that we use to optimize such energies, and their relation to the continuous problem. In section 5 we describe some specific models of regions and boundaries, and demonstrate the results of applying these models to images.

2. Related work

Early work on contour-based grouping includes Parent and Zucker's [25] work using relaxation methods, Sha'ashua and Ullman's [27] work on saliency networks, and Guy and Medioni's [10] work using voting schemes. Elder and Zucker [8] developed a method for finding closed contours using chains of tangent vectors, but they drastically prune the search space to render tractable the exponential problem they have set themselves. This work is based on edge maps rather than intensity gradients as such.

Mumford first pointed out the connection between minimal energy curves ("elastica") and stochastic processes [24], and discussed their application to computer vision. Williams, Thornber and Mahamud [21, 30] and Williams and Jacobs [31] find closed curves in edge maps using stochastic completion fields closely related to elastica. There is a close connection between their work and the minimum mean weight energy discussed in section 4, but this has not been fully explored. Closest to our work however, because it uses an energy optimization criterion explicitly, is the work on active contours and deformable models. The seminal work

in this area is Kass et al. [16] and Blake and Zisserman [3], and much subsequent work follows this both in the form of the energy functionals used, and in algorithmic techniques. Typical energies include data terms that favour contours passing through points of high intensity gradient magnitude, and regularizing terms that tend to keep the contour short or smooth. The energies used are linear however, and hence cannot exhibit scale invariance in the sense discussed in section 3.3. Linearity also means that theoretical obstacles prevent the global minimum from being both non-trivial and computable in polynomial time, so that gradient descent or a variant of it is normally used to find local minima starting from an initial position chosen by the user.

Another body of work applies dynamic programming techniques to minimize contour energies in an effort to reach global solutions, an approach that can be interpreted in terms of shortest paths in directed acyclic graphs. Amini et al. [2] use dynamic programming as part of a gradient descent procedure. Montanari [23] uses dynamic programming to find the minimum energy path between given end-points. Geiger et al. [9] use initialization with a series of points, and a choice of window around those points, to delineate the space of contours considered. In all cases initialization and restricted regions of the image are used to limit the space of contours over which the optimization proceeds, and in addition many of the algorithms find only local minima, or approximations to global minima over this limited set of contours. Globally minimum closed contours are not found.

The interesting paper of Zhu and Yuille [32] uses both region and boundary information within an energy optimization model for image segmentation. The work brings active contour and region growing techniques together within a Bayesian framework. The paper uses Green's theorem to compute the functional derivatives used in the gradient descent algorithms that form the core of the method. Because of the use of gradient descent however, the algorithm can find only local minima of the energy functional used, even in the case where only a single region is sought.

The paper by Cox, Rao, and Zhong [4] is particularly related to our work. They use a ratio energy also, of a generalised area to a generalised length, and use a "pinned ratio" algorithm to optimise it. This algorithm finds the optimal closed curve through a given point, and by trying successively all possible points in the image domain, the globally optimal curve can be found. Unfortunately, by its nature the algorithm is constrained to consider generalised areas, or in other words the integrals of positive functions on the interior of the region. This is very restrictive: in particular it means that the combination of region and boundary information that is possible in our model is not possible in theirs. In addition, the algorithm is rather unwieldy. Because of the way the algorithm works however, it is possible to restrict attention to those curves passing through a fixed point, which is not possible with the minimum ratio weight cycle algorithm described here.

Shi and Malik [28] use a generalized eigenvalue method to find normalized cuts of an image graph, and use this to partition the image by iterating the algorithm. The algorithm is a sophisticated and general clustering method, and as such, although Leung and Malik [20] extend the work interestingly by incorporating weak contour continuity information into the region-based model, does not seem amenable to the natural inclusion of boundary information. The big advantage of the normalised cut over previous minimum cut methods however is the normalisation, which re-scales the cut weight to remove trivial solutions associated with the removal of one or very few vertices. The denominator in our energy plays a very similar role to the cut normalization.

3. Theoretical framework

We will formulate the model in a continuous way, as this renders the properties of the energy functional manifest. In practice of course, the measured image function is defined on a graph G embedded in the image domain $D \subset \mathbb{R}^2$, G being typically, although not necessarily, a rectangular lattice. Hence the algorithms we use to optimize the energy functional will be formulated in the discrete domain, the continuous objects being approximated by their discrete counterparts: the continuous boundaries will be approximated by cycles in G for example. The choice to use a continuous model is justified by several considerations. The light intensity incident on the detector is described using continuous domains, and in fact this is the problem that we would really like to solve, hence the continual increase in the resolution of detectors. Second, the relation between the continuous and discrete formulations lies in the properties of the detector, and the use of both ways of phrasing the problem means that we can in principle study the effect of the detector on the algorithm. Third, the continuous formulation reveals the geometric structure and invariances of the model much more clearly than does a discrete formulation, which by its nature depends on the discretization used.

The form of energy functional we will describe is defined on the space of boundaries in D. By a *boundary*, denoted ∂R, we mean a closed curve in the image domain, under some restrictions that are important technically to ensure that all the expressions we will use are well-defined, but which are not that illuminating otherwise. We will allow self-intersections and repeated segments in the curves in principle but, as we discuss in section 4.1, such curves cannot be minimizers of the energy we will define except in degenerate cases. We will allow isolated corners in the curves where the tangent vector is not well-defined, because cycles in G have precisely this behaviour when viewed as continuous closed curves in D. We also include multiple closed curves, but again these only appear as minimizers of the energy in degenerate cases. If the situation were degenerate in one of the above ways, we would detect all the degenerate minima.

A boundary will be represented as an injection γ of the circle S^1 into D obeying the above restrictions. For any given boundary ∂R, there is an equivalence class of maps γ that correspond to it. They are related by bijections ϵ of the circle to itself obeying appropriate restrictions, so that γ is changed to $\gamma\epsilon$. Any energy functional should be invariant to such changes, because it is ∂R and not γ that contains the geometric information in which we are interested. If the functional is not invariant to such changes, the results will depend on an arbitrary choice made by the user or the algorithm.

The boundaries we use are *oriented*, or in other words they have a direction. The image domain D is also oriented, meaning that we can define the normal to any vector in the image domain (and in particular to the tangent vector of the boundary) as the rotation of the vector by $\pi/2$ in the direction of the orientation, for example clockwise. We denote the rotation of a vector v in this way $v^\perp$.

Having set up the background, we now write down the form of energy functional, and then explain the various terms in it.

$$E[\partial R] = \frac{N[\partial R]}{D[\partial R]} = \frac{\int_{S^1} dt\, \gamma'(t)^\perp \cdot v(\gamma(t))}{\int_{S^1} dt\, |\gamma'(t)|\, g(\gamma(t))} \tag{1}$$

We have already defined ∂R and γ, but note that $\gamma(t)$ is a point in D, and therefore stands for two coordinate values. t is an arbitrary parameterization of S^1, and a prime denotes a derivative with respect to t. $\gamma'(t)$ is thus the (un-normalised) tangent vector to the boundary,

and $\gamma'(t)^\perp$ is the (un-normalised) normal vector, oriented according to the orientation of the boundary. (We write the boundary integral this way rather than directly in terms of $\gamma'(t)$ because it simplifies the discussion of Green's theorem below, and gives a more intuitive picture of the energy.) We give the image domain the usual Euclidean metric, here denoted $\cdot$. The quantities v and g are where all the modelling takes place. The quantity v is any vector field on D, while g is any positive function. Both v and g will typically be derived in some way from the data, or will encode some geometric constraint. Note that N is dependent on the orientation of the boundary, changing sign under a change of orientation, whereas D is not. This means that values of E come in positive/negative pairs corresponding to the two orientations of a closed curve, and hence that minimizing E is the same as maximizing its absolute value. This is often a clearer way to view the functional.

Intuitively, the numerator $N[\partial R]$ measures the net 'flow' of some quantity v into or out of the region R that is the boundary's interior. In the most obvious case, which we use often, $v = \nabla I$, the intensity gradient. Then $N[\partial R]$ measures the flow of intensity into or out of the region, which is to say the total amount of intensity gradient normal to the boundary. The denominator is positive by construction, and hence can be viewed as a generalised measure of length. In this paper, we will almost always take the function g to be identically equal to 1, meaning that $D[\partial R]$ is just the Euclidean length. The meaning of the energy in equation 1 is then just the average flow per unit length into or out of the region.

It is easy to see that E is invariant to the choice of representative γ of a boundary ∂R: replacement of γ by $\gamma\epsilon$ leads to an identical expression after a change of variables in the integration. Equation 1 has other interesting invariances also, which we discuss in section 3.3.

3.1. **From region to boundary and back.** Equation 1 appears to deal solely with data on the boundary, but in practice we would also like to be able to say something about the region that is the interior of the boundary. For example, it would be useful to find the boundary that, in addition to having high intensity gradients along its length, contained a region of highly homogeneous intensity, or a particular texture. The remarkable thing is that equation 1 can incorporate such information, while still preserving the algorithmic property that it can be globally minimized in polynomial time. This is possible through the use of Green's theorem:

$$\int_{S^1} dt\, \gamma'(t)^\perp \cdot v(\gamma(t)) = \pm \int_R \nabla \cdot v(x, y)\, dx\, dy \tag{2}$$

This relates the integral of a vector field v over the boundary γ of a region R to an integral of its divergence $\nabla \cdot v$ over the interior. The sign on the right-hand side depends on the relative orientation of the boundary and the image domain. As it stands, equation 2 does not appear that useful, since it enables us to convert boundary data to region data rather than the reverse. However, since the image domain is a simply-connected subset of the plane, the reverse is also true. That is, given any function f on the image domain, the integral of this function over a region can always be converted to the integral of an associated vector field v_f over the boundary of the region. This is clearly possible if, for any function f, we can find a vector field v_f such that $\nabla \cdot v_f = f$. This is easily demonstrated. In components

with respect to rectangular coordinates x and y on D:

$$v_f{}^x(x,y) = \frac{1}{2}\int_a^x f(x',y)\,dx' \tag{3}$$
$$v_f{}^y(x,y) = \frac{1}{2}\int_b^y f(x,y')\,dy'$$

It is trivial to see that $\nabla \cdot v_f = f$. There is a lot of freedom in choosing v_f. The choice of a and b does not affect the divergence, and neither does the replacement of the factors of $\frac{1}{2}$ by two numbers p and q such that $p+q=1$. The addition to v_f^x of a function depending solely on y and the addition to v_f^y of a function depending solely on x also have no effect. These are special cases of the fact that one can add to v any divergence-free vector field u and still obtain the same function when the divergence is taken. This property is discussed further in section 3.3. Note that the vector field is easy to evaluate algorithmically. One simply traverses the image once along each row and once along each column, adding up the values of f as one goes along. For maximum efficiency, the y-component can be eliminated entirely by choosing $p=1$ and $q=0$, thus making only one image traversal necessary.

In the special but important case that f is produced by the convolution of some kernel with the intensity, the vector field is even easier to derive. We have that

$$f(x,y) = \int_D K(x,y,x',y')\,I(x',y')\,dx'\,dy' \tag{4}$$

Then we have for the case of $v_f{}^x$:

$$\begin{aligned} v_f{}^x(x,y) &= \frac{1}{2}\int_a^x \left[\int_D K(x',y,x'',y'')\,I(x'',y'')\,dx''\,dy''\right] dx' \\ &= \frac{1}{2}\int_D \left[\int_a^x K(x',y,x'',y'')\,dx'\right] I(x'',y'')\,dx''\,dy'' \\ &= \frac{1}{2}\int_D K^x(x,y,x'',y'')\,I(x'',y'')\,dx''\,dy'' \end{aligned} \tag{5}$$

and similarly for $v_f{}^y$. Thus if the kernel is well enough behaved to allow the exchange of order of integration, then rather than having to perform the integrations in equations 3 for each image that we process (although this is scarcely time-consuming), we can integrate the kernel once and for all.

The above discussion means that *any* measure of region optimality expressible as an integral over the region of some function can be re-expressed as an integral over its boundary. Given some measure of boundary optimality u and some measure of region optimality f, we can simply form the vector field $u+v_f$ and use it in equation 1. Thus equation 1 can include region information of this type as easily as it can boundary information such as intensity gradients. Note however that as soon as we start forming linear combinations of different sources of information, we introduce parameters into the model representing the relative weight to be given to these different sources.

3.2. **Examples of vector fields and functions.** To clarify these abstract ideas, we give some examples of functions that can be built from the image intensity and used in equation 1. We give the examples in terms of functions integrated over the region, and where analytical evaluation of equations 3 is possible, we also indicate the corresponding boundary vector field. We denote convolution by $*$.

The most obvious possibility is to choose $f = I$. In this case the model is looking for high intensity regions. It will find bright spots such as specular reflections, as well as large regions of high intensity. Similarly, $f = e^{-I}$ would look for dark regions, and in general f can be designed to look for regions of any particular intensity.

Another obvious choice is $f = 1$. In this case, the region integral is measuring the area. The vector field v_f is then just given by ${v_f}^x(x, y) = \frac{1}{2}x$, ${v_f}^y(x, y) = \frac{1}{2}y$. Including this term then favours regions of larger area.

Another useful choice, which we have already mentioned, is $f = \nabla^2 I$. This corresponds to a vector field $v_f = \nabla I$. The form of the numerator of equation 1 means that boundaries that lie on high intensity gradients in the image and that are oriented so that the gradient is normal to the boundary will be favoured by this choice. Compared to the standard active contour energy, this choice has the advantage of favouring gradients normal to the boundary. In some situations however, this choice of f does not perform as we would wish. Note that it favours boundaries in which the gradient is consistently oriented towards the interior or the exterior of the region. It does not therefore model contrast-reversing boundaries well. We describe below a way to deal with this eventuality via a different choice for the function g.

Choosing f to be a monotonically-decreasing function of the intensity gradient magnitude, for example $f = e^{-|\nabla I|}$, will favour regions that have a homogeneous intensity across them.

We now describe a general form for f that can be used for example with texture descriptors and in many other cases. Suppose we have some map T that, given an image I, generates a map from the image domain to some feature space $\mathcal{F}$: $D \overset{T[I]}{\to} \mathcal{F}$. We assume $\mathcal{F}$ to be a metric space with metric ρ. Now given a point (or in general a subset) $s \in \mathcal{F}$, we can define a function on D as

$$f(x, y) = M(\rho(T[I](x, y), s)) \tag{6}$$

where M is a monotonically-decreasing function of its argument. The value $f(x, y)$ is thus large when the feature value at point (x, y), $T[I](x, y)$, is close to s. The integral of f over a region will thus be large if the region has feature values close to s. The previous examples of functions f are all special cases of this general formulation. Typical examples of $\mathcal{F}$ are one or another colour space, or the space of values taken by a set of filters designed to describe texture.

Combining various terms in the numerator raises an issue beyond the addition of parameters. An easy way to illustrate it is to take as an example the combination $f = \nabla^2 I + 1$. This means that f favours regions with high intensity gradient normal to the boundary and with larger area. The situation is not quite that simple though, due to the relation between the orientation of the image domain and the orientation of the boundary. For a particular fixed choice for the orientation of D, such a function will favour larger area regions with the gradient oriented outwards ('a black blob on a white background'), while the function $\bar{f} = \nabla^2 I - 1$ will favour larger area regions with the gradient oriented inwards ('a white blob on a black background'). If our modelling requirements are that these two situations be considered equivalent, then both signs must be used and the energies compared.

We have not discussed so far any possibilities for the function g appearing in the denominator of equation 1. For most of the paper we assume that $g \equiv 1$, so that $D[\partial R]$ is the Euclidean length. However, as mentioned above, to deal with contrast-reversing boundaries it is necessary to alter g. By putting $g = |\nabla I|^{-1}$ (or some other decreasing function), we

favour boundaries passing through points of high intensity gradient. We still have to choose a term for the numerator, and here the most neutral choice is the area, $f = 1$. We thus favour larger area regions with a great deal of gradient on the boundary, regardless of orientation.

3.3. **Invariances.** Equation 1 has some interesting invariances beyond the necessity that it be invariant to the choice of boundary representative γ. Adding to v another vector field u, we can use Green's theorem to convert the integral of u over the boundary to an integral of its divergence $\nabla \cdot u$ over the interior R. Then if $\nabla \cdot u = 0$, $N[\partial R]$ will be unchanged by the addition of u to v. If $v = \nabla I$, then the addition to I of another function i will not affect the boundary integral if $\nabla \cdot \nabla i = \nabla^2 i = 0$. So changing the image by the addition of a harmonic function will not change the output of the model. Note that this includes the addition to the intensity of a constant, or of a linear function.

As mentioned in section 1, the energy E may possess a 'scale invariance'. By this we do not refer to invariance to changes in the size of the image (or equivalently to changes in the metric on the image domain), but to the relation between the energy of a boundary and a scaled version of itself. Clearly this depends on the data that enters the energy via the quantities v and g. If γ is an arbitrary boundary, and $\mu\gamma$ is a scaled version of it, then the change from γ to $\mu\gamma$ introduces a factor of μ into both numerator and denominator from the terms in γ'. These factors therefore cancel. If the vector field and function are unchanged, which is to say that

$$v(\gamma(t)) = v(\mu\gamma(t)) \qquad g(\gamma(t)) = g(\mu\gamma(t)) \tag{7}$$

for all t, then the energy of the original and scaled boundaries will be the same. Thus the energy is scale-invariant if the data under the boundary is the same, which is as one would expect. Note that this is close to the idea of invariance to object size. In an image, the boundaries of a large car will exhibit almost the same behaviour as those of a small car. In the case that $v = \nabla I$ and $g \equiv 1$, the above equations are satisfied if the gradient of the intensity is the same on the scaled boundary as on the original, which justifies calling this and similar instances of the energy scale-invariant. On the other hand, it is clear that if v is the integral of a constant function, $v \propto (x, y)$, so that its integral around the boundary gives the area of the interior, then equation 7 cannot be satisfied.

The latter behaviour does not represent a problem. The point of any invariance is that it should be obeyed in the absence of any modelling decision to the contrary. The introduction of an area term certainly constitutes a decision to break the invariance. The difference between the energy in equation 1 and standard linear active contour energies is that equation 1 *can* be free of bias towards small or large boundaries if desired. Linear energies cannot, so that breaking the invariance is unavoidable. The question that remains then is: what are the vector fields and functions that constitute a decision to violate scale invariance?

This is a hard question to answer in the abstract. The vector field v and function g can in principle be arbitrarily complicated non-local functions $v[I]$ and $g[I]$ of the intensity. We then have that v maintains scale invariance if for all γ, μ and u, there is an image I such that $v[I](\mu\gamma(t)) = u(t)$, where u is the data vector field on the original boundary γ. A similar equation holds for g. In short, those vector fields violate scale invariance that cannot be made to satisfy equation 7, in the sense that if they did there would be no image from which they could have been derived. Various special cases can be analysed, such as when

v is derived from I by a linear map, but space requirements prevent full discussion of these here.

4. Algorithmics

After the long discussion of the theoretical properties of the model, we now turn to the algorithms that will enable us to solve it. The first algorithm applies in a restricted set of instances, but has the advantage of being extremely parallelizable. The second is completely general, applying to any energy functional of the form of equation 1, and on serial machines is much faster than the first algorithm.[1] Neither algorithm requires initialization, meaning that given an image, the algorithm outputs a boundary without the user having to interfere by selecting a starting contour or a number of seeds. This does not mean that it is impossible for the user to control the algorithm: he can select regions of the image in which to search, for example an annulus around an interesting boundary, or he could add potentials to the data that are incorporated into the model and that direct the algorithm.

In subsection 4.1, we describe the discrete problem that we will solve. Then, in sections 4.2 and 4.3, we describe the two algorithms that solve the problem and compare them. In subsection 4.4 we describe the relation of the discrete problem to the continuous one.

4.1. **Problem.** Given a graph $G = \langle V, E \rangle$ (we denote $|V|$ by n, and $|E|$ by m), and two maps $\lambda : E \to \mathbb{Z}$ and $\tau : E \to \mathbb{Z}^+$, we define the 'ratio weight' $W(C)$ of a set of edges $C \subset E$ as

$$W(C) = \frac{\sum_{e \in C} \lambda(e)}{\sum_{e \in C} \tau(e)} \tag{8}$$

Note that the weights are integral: this represents almost no restriction in practice. Note also that the co-domain of τ is the positive integers. This corresponds to the form of the integrand in $D[\partial R]$ in equation 1, which is positive by definition. When τ is constant, we call W the 'mean weight.' Let $\mathcal{C}$ be the set of cycles in G. The problem is then to find $W^* = \min_{C \in \mathcal{C}} W(C)$ and $C^* = \arg\min_{C \in \mathcal{C}} W(C)$. We will call this problem A.

There is another problem, problem B, where we wish to find the minimum 'total weight' cycle with weight given by a single sum like the numerator of equation 8. Problem B is the discrete equivalent of the minimization problems arising from linear active contour energies, and comparison of the two problems in both the continuous and discrete domains shows how the properties of the energy in equation 1 arise. In the continuous domain, note the following points. If the linear energy is never negative, for example if it is orientation-independent like the denominator of equation 1, then attempts to minimize it will simply lead to the trivial solution of an infinitely small boundary with zero energy. If the linear energy is orientation-dependent, like the numerator in equation 1, then there will be a boundary with negative energy unless all boundaries have zero energy. This means that the energy is unbounded below, because a boundary can wrap around a negative energy boundary an infinite number of times. Since the addition of any other boundary to this boundary will not change the infinitely negative energy, the problem is ill-posed. Thus the global solutions in these two cases are trivial.

The same conclusions arise in the discrete case, but here there is a third possibility: an energy that is neither orientation-independent (an undirected graph) nor orientation-dependent (a directed graph in which opposing edges have weights of opposite sign). Although there

[1] There is also a linear programming approach to the problem, described by Dantzig, Blatner and Rao [5].

is no way to conclude that there is a negative cycle in this more general case, a problem still arises. Minimizing cycles may self-intersect or even self-overlap, neither of which is very satisfactory. One can solve these problems simply by deciding to minimize over non-self-intersecting boundaries only. Unfortunately, although theoretically possible, this restriction creates an NP-hard problem. This is because a polynomial solution of problem B restricted to simple cycles would allow a solution of the Hamiltonian cycle problem in polynomial time.

Linear energies thus lead us to an impasse: either the global solution is trivial or the problem is essentially insoluble. How does the energy in equation 1 avoid these problems? Note first that the 'wrap-around' problem does not arise, since an infinitely-wrapped boundary will have exactly the same energy as a singly wrapped one. The same phenomenon also takes care of self-intersecting and self-overlapping boundaries/cycles. Self-intersections are eliminated as follows. Suppose for specificity that the boundary is a figure of eight. If the loops have different energies under equation 1 when considered as separate boundaries, then one of them will have a lower energy than the other. In that case, the combined figure of eight boundary will have a higher energy than the loop with lower energy, and therefore cannot be a global minimum. If the loops happen to have exactly the same energy considered separately, then the situation is degenerate, and the algorithm we will describe would find both loops of the figure of eight. They can then trivially be separated as equivalent global minima. The case of self-overlaps is a little more subtle, but in the orientation-dependent case at hand things simplify, and we can eliminate self-overlaps from consideration via the following argument. If a boundary self-overlaps, then the overlapping section will add nothing to the numerator of equation 1, but will increase the denominator, thereby bringing down the absolute value of the energy. Such boundaries cannot therefore be minima of equation 1 since removal of the overlapping section would produce a boundary with a lower energy.

We now proceed to describe the algorithms.

4.2. **Minimum Mean Weight Algorithm.** This algorithm does not solve the general instance of problem A. The restriction is that the denominator weights τ should all be equal, or in other words that we are solving the 'minimum mean weight cycle' problem rather than the more general 'minimum ratio weight cycle' problem. Up to an irrelevant factor, $\tau \equiv 1$, so the denominator is simply counting the edges in the set $\mathcal{C}$. This discrete problem does not have a continuous counterpart, dependent as it is on the discretization. The algorithm is due to Richard Karp, and we refer the reader to the original paper for proofs [15].

First, fix an arbitrary start vertex $s \in V$ and define the function F_k taking each vertex $v \in V$ to the minimum of the total weight (defined using λ) over the set of paths to v from s that consist of exactly $k \geq 0$ edges. If no path of k edges exists, then define the weight to be ∞. Then it can be shown that the weight W^* of the minimum mean weight cycle is given by

$$W^* = \min_{v \in V} \max_{k \in [0..(n-1)]} \left\{ \frac{F_n(v) - F_k(v)}{n-k} \right\}. \tag{9}$$

$F_k(v)$ can be computed using the recurrence

$$\begin{aligned} F_k(v) &= \min_{(u,v) \in E} F_{k-1}(u) + \lambda(u,v) \\ F_0(s) &= 0 \\ F_0(v) &= \infty \ , \ \forall v \neq s. \end{aligned} \tag{10}$$

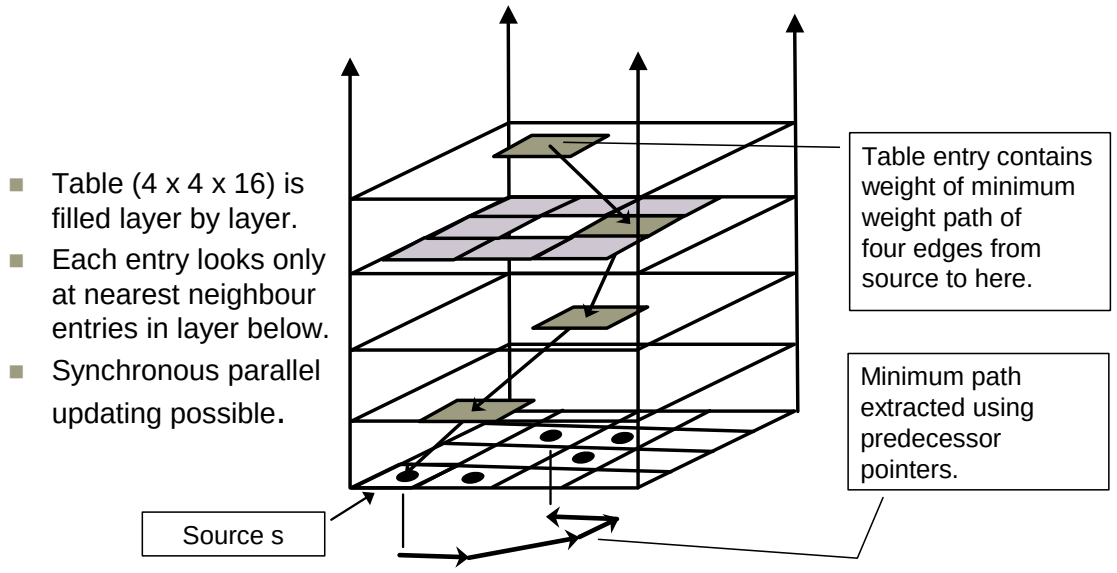

FIGURE 1. The figure illustrates the dynamic programming that takes place in the solution of the minimum mean weight cycle problem. An empty table is constructed, with the graph vertices as base, and with height equal to the number of vertices. The lowest level is then filled in using the second and third of equations 10. Subsequent levels are filled using the first of equations 10. By recording the minimizing vertices at each level, the minimum mean weight cycles can be found as described in the text. The intuition is that as the paths grow up the table, they start to wrap around low weight cycles. Equation 9 then 'cuts off' the beginning of the path from the source to the cycle, and evaluates the mean weight of the cycle that is left.

The computation of F for all $k \in [0..(n-1)]$ can be performed using dynamic programming in time $O(nm)$. This process is illustrated in figure 1. A table with the graph vertices as a base, and with height equal to the number of vertices in the graph, is filled layer by layer using equations 10. The minimum weight paths can be tracked simultaneously by recording, each time it is used, which vertex minimizes the first of equations 10. This data then gives a predecessor graph for each minimizing path from s.

Having filled the table, we can compute W^* from $F_k(v)$ by finding the maxima and minimum in equation 9 in a further $O(n^2)$ time, leading to an overall computation time of $O(nm)$. Clearly the table requires $O(n^2)$ of memory. The cycle itself can be extracted by recording the minimizing v and k in equation 9, and then finding a cycle of length $n-k$ in the minimum weight path from s to v. The latter can be done simply by back-tracking in the predecessor graph from v.

Since $F_k(v)$ only depends on $F_{k-1}(u)$ for u in the neighbourhood of v, a network arranged in levels of constant k could compute the values of F_k in parallel from the values of F_{k-1} in $O(1)$ time. To fill the whole table containing $F_k(v)$ for all k and v would thus take $O(n)$ time. The minimization would take the same time, and is also easily implementable as part of the network. It is easy to envisage columns of this sort being arranged behind a detector and producing at the 'top' of the table the extracted region.

4.3. **Minimum Ratio Weight Algorithm.** The minimum mean weight cycle algorithm described above has the drawback that it cannot deal with edge weights τ other than the trivial case in which $\tau \equiv 1$. This has the consequence that it cannot deal with arbitrary functions g in $D[\partial R]$ in equation 1. Indeed, it cannot deal even with the case of the function $g \equiv 1$, since this corresponds to $\tau(e) =$ Euclidean length of e. Instead it uses a discrete measure: the number of edges in the discretized version of the boundary. This lack of generality and dependence on the discretization, coupled with the unfortunate properties of an edge count as a measure of distance, render the algorithm unsatisfactory for many purposes. The minimum ratio weight cycle algorithm described in this subsection works for arbitrary $\tau : E \to \mathbb{Z}^+$, and hence deals with arbitrary functions g in $D[\partial R]$. It also requires considerably less memory resources than the minimum mean weight cycle algorithm. The main consequence of this parsimony with memory is increased speed. Instead of the execution times of a few minutes found for Karp's algorithm, we find execution times on single images of a few seconds (typically less than five seconds on a 256×256 image) for the minimum ratio weight algorithm.The algorithm has the added advantage that degenerate minima of the energy 1 can be identified simultaneously.

The algorithm was first described by Lawler [19], and since then has been much generalized by Nimrod Meggido [22]. It relies on the following, interesting observation. We can define a new, parameterized edge weight $E \xrightarrow{w_t} \mathbb{Q} : w_t(e) = \lambda(e) - t\tau(e)$, where $t \in \mathbb{Q}$. We use the same symbol w for the weight of a set of edges (defined by summation). Then the solution, t^*, of $w_t(C_t^*) = 0$, where C_t^* is the solution to problem B with weights w_t, is equal to the minimum ratio weight W^* in problem A, and the minimizing cycle $C_{t^*}^*$ of problem B is equal to C^* for problem A. The simple proof is given in appendix A.

The problem is thus reduced to finding t^*, or in other words the value of t such that the minimum total weight cycle using w_t has zero weight. This in turn can be found by finding the largest value of t such that G weighted by w_t has no negative cycle. Although there are a number of approaches, including binary search [19], and a more sophisticated approach using parametric edge weights [22], we find in practice that the fastest method is simply linear search. The algorithm is shown in figure 2. We assume that the object `G` corresponding to the graph G contains the edge weights λ and τ, and so only needs t to compute w_t. The function call `existsNegativeCycle(G,t,X)` returns `true` if a negative cycle exists in graph `G` with weights w_t, and sets `X` equal to its ratio weight. It returns `false` if there is no negative cycle. The function call `findZeroCycle(G,t)` returns the (one or more) zero weight cycles in the graph `G` with weights w_t. Appendix A contains details of the negative and zero cycle detection algorithms that we use.

To see how this works, note that if $t > t^*$, the graph G using the weights w_t will have a negative cycle. We therefore start with a known upper bound t^0 on t^*. We apply a negative cycle detection algorithm with edge weights w_{t^0}. If we do not find a negative cycle then there is necessarily a zero weight cycle, and by the proof in the appendix we are done: $t^0 = t^* = W^*$. The zero weight cycle $C_{t^0}^*$ is a solution C^* to problem A (as are any other zero weight cycles—hence we can detect degenerate minima). On the other hand, if a negative cycle C *is* detected, then t^0 is too large. Since $w_{t^0}(C) < 0$, we have that $t^0 > W(C) \geq t^*$. We therefore replace t^0 by $t^1 = W(C)$. The search continues in this fashion until there is no negative cycle, at which point there must be a zero cycle $C_{t^*}^*$ with ratio weight $W^* = t^*$ and $C^* = C_{t^*}^*$.

```
t = t0; // t0 is a known upper bound on t*
while (existsNegativeCycle(G,t,X))
    t = X;
return findZeroCycle(G,t);
```

FIGURE 2. Pseudo-code for the Minimum Ratio Weight Cycle Algorithm.

Note that because the weights λ and τ are integral, t^* is rational, as already stated. This enables a pseudo-polynomial bound to be placed on the search time in the following way. It is easy to see that because of the integrality of the weights, upper and lower bounds on W^* are given by $\pm\lambda_0$, where $\lambda_0 = \max\{\lambda(e) : e \in E\}$. The minimum weight difference between two distinct ratio weights can similarly be proved to be $\frac{1}{\tau_0^2}$, where $\tau_0 = \max\{\tau(e) : e \in E\}$. Thus the maximum number of applications of the negative cycle detection algorithm is $O(\tau_0^2\lambda_0)$. Since the asymptotic complexity of negative cycle detection is $O(mn)$ for the algorithm we use, the pseudo-polynomial bound follows. In practice, the negative cycle detection never executes to completion and the time bound is never saturated. The algorithm requires $O(m)$ space, much less than Karp's algorithm. Further details of the time and space bounds are to be found in appendix A.

There are also close to linear time negative cycle detection algorithms for planar graphs [17]. These would allow improved performance in the case of single images, but we prefer to describe the general case here since it generalises to higher dimensions, the application of which to stereo and motion is described in [11].

Note that the way the algorithm works shows what was claimed in section 1, that the energy in equation 1 is equivalent (as far as global minima go) to a linear energy of the form $N[\partial R] - \beta D[\partial R]$, but with β chosen automatically to be equal to the global minimum of equation 1. The parameter is thus a complicated functional of the image.

4.4. **Application.** In order to solve the continuous problem of finding the minimum of equation 1, we must show how it is related to the discrete problem described in section 4.1.

To do this, we embed a graph G in the image domain. The edges in the graph now correspond to line segments in the image domain, and cycles in the graph correspond to boundaries in the image domain of the type described at the beginning of section 3. If we denote the integrands in $N[\partial R]$ and $D[\partial R]$ by n and d respectively, we can define weights λ and τ for each edge e by integrating n and d along the line segment corresponding to e. Because $N[\partial R]$ and $D[\partial R]$ are both linear functionals on the space of boundaries, their values on a boundary ∂R in D corresponding to a cycle C in G are given by the sums over the edges in C of λ and τ. If we restrict our attention to the boundaries in D that correspond to cycles in G, then there is an exact correspondence between the continuous energy of equation 1 and the discrete ratio weight of equation 8. We can therefore exactly solve the continuous problem on this restricted space of boundaries. If the graph is embedded densely enough in D, this in its turn will be an approximation to the solution of the continuous minimization problem over the full space of boundaries in the image domain. This is a heuristic statement: the characterization of the accuracy of this approximation is a difficult problem for which we do not have a solution. This is connected to the nature of the detector used to acquire the image, as well as to the properties of the continuous signal itself.

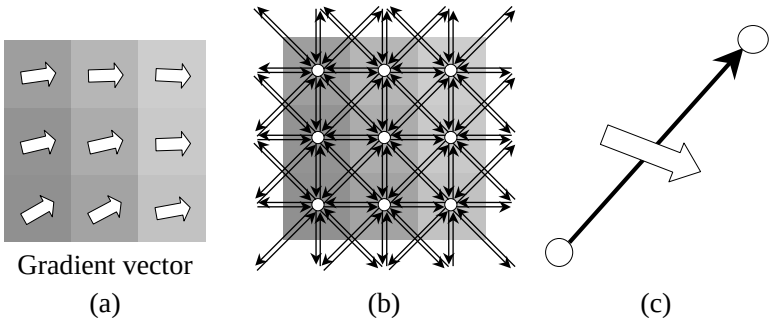

FIGURE 3. (a) For each pixel we compute the gradient vector. (b) The graph has a vertex for each pixel and eight outgoing edges for each vertex (except at the boundary.) (c) The edge weight is calculated by taking a cross product of the gradient vector and the edge vector.

In practice, we choose the graph to have as vertices the pixels in the image, and as out edges the eight neighbours of each pixel. This is shown in part (b) of figure 3. We then must evaluate λ and τ for each edge. This is done as follows. Given an edge e, we have the in- (i) and out- (o) vertices (pixels), viewed as points in $D \subset \mathbb{R}^2$. The tangent vector along each edge is simply $o - i$. We define v and e on each edge as the average of their values at i and o. This corresponds to replacing v and g by linear approximations between pixels and then integrating n and d. This gives us all the data we need to evaluate λ and τ. Note that this construction does not depend on the details of v or g: it can be carried out for any energy in the form of equation 1.

5. DEMONSTRATIONS

5.1. **Intensity gradient and area.** For the first set of experiments, we chose $f = \nabla^2 I \pm \beta$, where β is a constant. As discussed above, the Laplacian gives a boundary term $v_f = \nabla I$, which favours boundaries with a high normal component of intensity gradient, while the second term favours larger areas. Its purpose is to eliminate very small regions. We took $\beta = 10/3$. We took the function g in the denominator of equation 1 to be identically equal to 1, thus measuring the boundary length. The energy in equation 1 then becomes

$$E[\partial R] = \frac{\int_{S^1} dt\, \gamma'(t)^\perp \cdot \nabla I(\gamma(t))}{\int_{S^1} dt\, |\gamma'(t)|} + \frac{\int_{S^1} dt\, \gamma'(t)^\perp \cdot u_\beta(\gamma(t))}{\int_{S^1} dt\, |\gamma'(t)|}, \tag{11}$$

where $u_\beta = (\frac{1}{2}\beta x, \frac{1}{2}\beta y)$ is the vector field obtained by integrating the constant function $f(x, y) = \beta$ according to equation 3.

To calculate the first term of the numerator in equation 11, we did the following. At each vertex/pixel, we computed the discrete version B of the gradient vector field ∇I by taking a Gaussian derivative on a scale of the order of a pixel:

$$B = \nabla G * I \tag{12}$$

Following the general prescription of the last section, we computed the numerator edge weight for an edge with in-vertex i and out-vertex o as the signed magnitude of the cross product of the vector $o - i$ with the average of the gradients at o and i: $\frac{1}{2}(B(o) + B(i))$. The cross product produces the same effect as rotating the tangent vector by $\pi/2$ and then

taking the dot product. The edge weight for the denominator we took to be its geometric length $|o - i|^{1/2}$, corresponding to our choice of $g \equiv 1$. This is illustrated in figure 3.

We applied the algorithm explained in section 4.3 several times to each image. After each iteration, we removed from the graph those vertices through which the previous solution had passed. This has the effect of eliminating a great deal of confusing repetition in which the algorithm finds boundaries that are nearly but not quite the same. In this way a series of regions of increasing energy was extracted. (Although this procedure does not necessarily disconnect the graph, it has much the same effect, meaning that overlapping regions are unlikely to be found. An alternative is to remove the edges in the cycle but not the vertices. This allows overlapping regions, while still tending to eliminate near repetitions.) The results are shown in figure 4. The numbers indicate the order in which the regions were found.[2] For the size of images used here, detection of one contour generally takes a few seconds to a minute on a 933MHz Pentium III computer. It uses about 15 Megabytes of memory for a 256×256 image.

The model is biased towards larger areas, but some small areas are still found. This is because they are 'spikes' in the intensity function. With a smaller value of β, more small areas are found, but the larger regions displayed are still among the first few discovered. The appearance of small regions for small values of β is not a defect of the model. It is the result of using a scale-invariant theory. As discussed in section 3.3, there is no reason *a priori* to prefer large regions to small. Only the task at hand can determine which is more important, and hence this should be a modeling decision rather than an unavoidable property of the energy.

5.2. **Contrast-reversing boundaries.** The energy used in the previous subsection favours boundaries for which the intensity gradient is oriented consistently inwards or outwards along its length. This means that it will not find a contrast-reversing boundary. In order to illustrate that the model can also deal with the case of contrast-reversing boundaries, we use a different energy functional, described in section 3.2:

$$E[\partial R] = \frac{\int_{S^1} dt\, \gamma'(t)^{\perp} \cdot u_\beta(\gamma(t))}{\int_{S^1} dt\, |\gamma'(t)|\, |\nabla I(\gamma(t))|^{-1}}. \tag{13}$$

The numerator now consists only of the area term. The denominator in turn has the role of detecting boundaries passing through high intensity gradient points in the image. The results of applying the two energy functionals to a synthetic contrast-reversing boundary are shown in figure 5. This energy also works on real images. Some results are shown in figure 6.

5.3. **Curves that intersect the image domain boundary.** Since the algorithm detects closed curves, it cannot detect a curve that intersects the image domain boundary. However,

[2]We show multiple regions out of interest, rather than because we believe that the algorithm is useful for image domain partition. We view the current model more as a base for the development of more sophisticated region modelling techniques. Nevertheless the question arises as to how many regions one should find in this way. We regard this question as ill-posed. If the algorithm were to claim to be finding the regions corresponding to some semantic entity, for example human beings, then there would be a well-defined way in which it would be right or wrong to find a certain number of regions. Clearly no such claim, or rather no remotely successful such claim, can be made for the algorithm. In the absence of such a semantic interpretation, the question of number of regions has no testable answer. These considerations apply to almost all, if not all segmentation and region identification algorithms, and not merely the one under discussion.

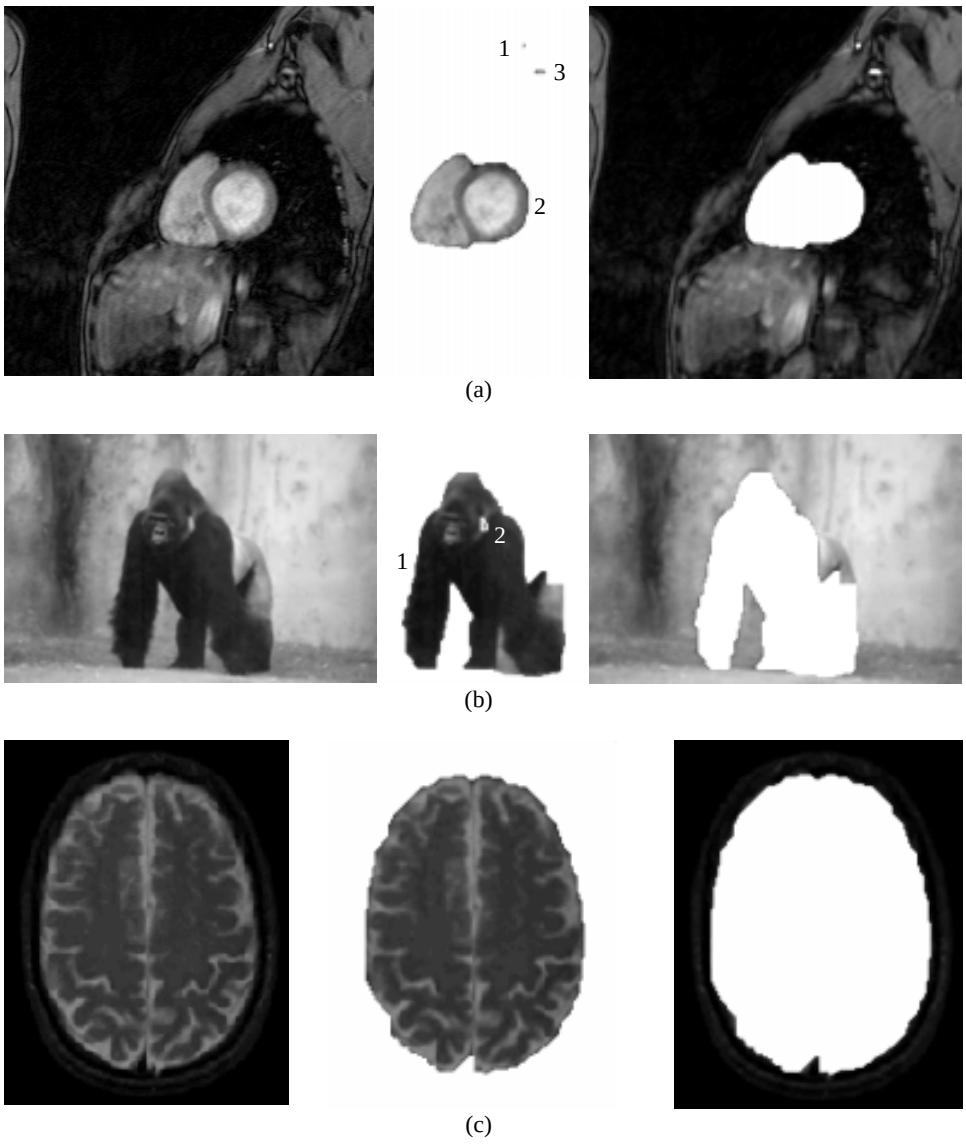

Figure 4. The result of applying the energy functional equation 11. (a) A 256×256 pixel image. Three regions are shown. (b) A 200×134 pixel image. Shown are two least energy regions. (c) A 124×166 pixel image. The best region is shown.

there is a simple remedy to this problem. We add to the graph a special vertex and connect it to every vertex corresponding to an image domain boundary pixel. We introduce edges in both directions, and give each edge zero λ-weight and a τ-weight of 1. This has the effect of connecting any pair of image domain boundary pixels by two edges that have little effect on the functional. A curve that intersects the image domain boundary can now be described as a boundary consisting of the portion of the curve in the image domain, plus the two edges that join the points of intersection with the image domain boundary. The result of using this graph with the gradient energy of equation 11 is shown in figure 7. There is a caveat though. In continuous terms, this procedure corresponds to changing the topology of the image domain into a sphere by identifying all points on its boundary. The result is that not

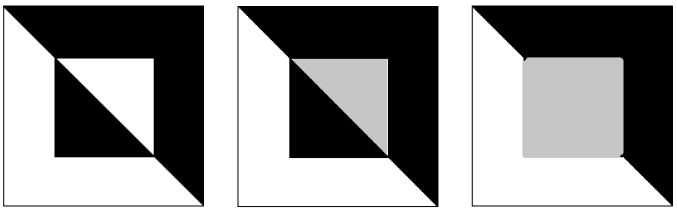

FIGURE 5. (a) A synthetic contrast-reversing boundary. (b) The result of applying the energy functional equation 11. (c) The result of applying the energy functional equation 13. The region found is shown in grey.

all functions can be integrated globally to give a vector field. Specifically, the area term does not work with this topology. Thus, the energy functional used for this experiment is only the first term of equation 11, which is defined as a vector field *a priori*.

6. SUMMARY

While the method we have presented is a powerful one, it has some limitations. Neither algorithm is 'interactive', in the sense that one can move the solution to where one would like it to be: this is a property of any algorithm that finds a global optimum. It is trivial however to restrict the search to parts of the image domain, and even to introduce potentials, similar to the 'volcanoes' used in the original active contour paper [16], to direct the search. If run-time interactivity is desired, then it is easy enough to sacrifice global optimality to interactivity by using an algorithmic approach more akin to gradient descent. None of the desirable theoretical properties of equation 1 would be lost. However, we view the energy as a foundation on which more sophisticated models of regions can be built, so that global optimality has the advantage of being fully automatic and of giving a clear picture of the significance of the results for the image as a whole.

One limitation of the method is that it cannot detect multiply-connected regions, for example annuli, except in degenerate cases. How serious a limitation this is depends again on the uses to which one sees the method being put. Segmentation algorithms may require multiply-connected regions, since they are not looking for isolated objects but a global partition, although in fact they are usually powerless to control things one way or the other. Specific objects in images however generally occupy simply-connected regions (completions are included here), and if they do not, what is usually needed is rather a self-intersecting boundary (think of the silhouette of a human being with his hands on his hips). This is because the usual reason that objects occupy multiply-connected regions in images is because they are the projections of volumes.

To summarize: the model we have described is one of few where the ability to find the global optimum does not restrict the range of modelling possibilities available. The energy allows the use of arbitrary (first-order) region and boundary information, while still being globally optimizable in polynomial time for any choice of this information using the same graph algorithm. Other theoretical properties of the model are good, allowing control over the scaling of the energy with respect to boundary size, if desired eliminating the uncontrolled

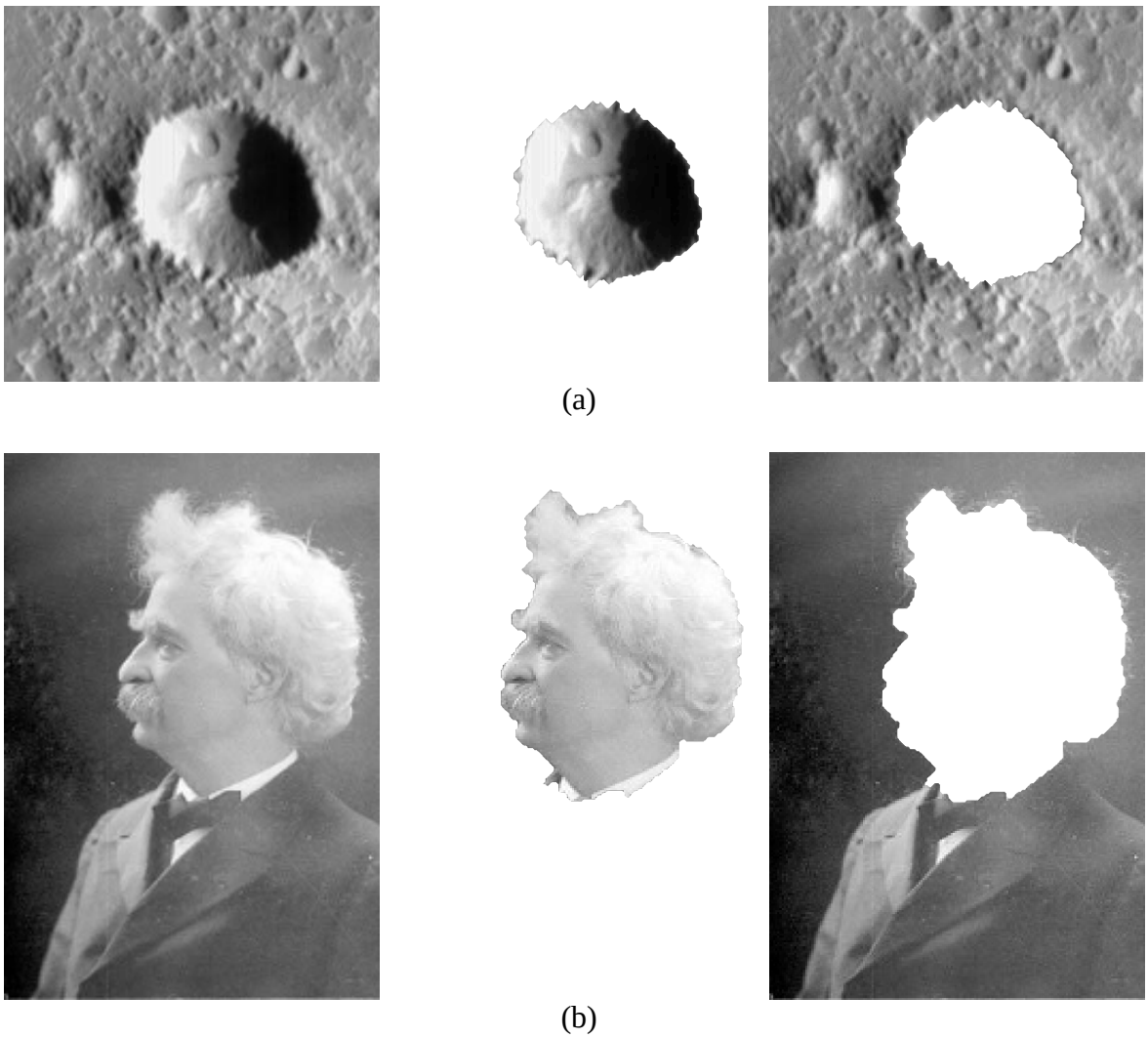

FIGURE 6. The results of applying the energy functional equation 13 to real images. (a) A 256 × 256 pixel image. Note that the extracted boundary is contrast reversing, i.e., for the left-hand-side boundary of the crater, the inside is lighter, while for the right hand side the inside is darker. (b) The same energy can also detect a non-contrast-reversing boundary (176 × 256 pixel image).

bias towards small or large boundaries present in linear active contour models. The algorithms themselves are fast and elegant, and special cases that allow improved performance are known. We are currently investigating further modelling possibilities of the energy.

Acknowledgements. The authors wish to thank the Instituto de Matemática Pura e Aplicada in Rio de Janeiro for their generous hospitality, and Professor Davi Geiger for his advice and encouragement. This work was partially supported by grant number MURI-AFOSR 25-74100-F1837.

FIGURE 7. A 176 × 256 pixel image. A curve that intersects the image domain boundary is detected by using equation 11 and a special vertex that is connected to all image domain boundary vertices.

APPENDIX A. ALGORITHMIC DETAILS

The claim in section 4.3 about the relation between the minimum ratio weight cycle problem and a related minimum weight cycle problem is proved as follows.

Proof. Suppose t^* is the solution to $w_t(C_t^*)$, where C_t^* is the minimum total weight cycle for the weight w_t. Then we have by definition that $\lambda(C_{t^*}^*) - t^*\tau(C_{t^*}^*) = 0$, and hence that $t^* = \frac{\lambda(C_{t^*}^*)}{\tau(C_{t^*}^*)}$. The claim is that t^* is the minimum ratio weight cycle for $W(C) = \frac{\lambda(C)}{\tau(C)}$, and that therefore $C_{t^*}^*$ is a minimizing cycle. Suppose this were not the case. Then there must exist a cycle C such that $t = \frac{\lambda(C)}{\tau(C)} < t^*$. This however would mean that $w_{t^*}(C) = \lambda(C) - t^*\tau(C) < 0$ or in other words that $w_{t^*}(C) < w_{t^*}(C_{t^*}^*)$, contradicting the assumed minimality of $C_{t^*}^*$.

For the reverse argument, suppose that t^* is the minimum ratio weight for $W(C) = \frac{\lambda(C)}{\tau(C)}$ and that C^* is a minimizing cycle. Then by definition, $t^* = \frac{\lambda(C^*)}{\tau(C^*)}$, or in other words, $w_{t^*}(C^*) = \lambda(C^*) - t^*\tau(C^*) = 0$. Now the claim is that C^* is a minimum total weight cycle for weight w_{t^*}, $C_{t^*}^*$, and that its weight is zero: $w_{t^*}(C^*) = 0$. Suppose this were not the case. Then there must exist a cycle C such that $w_{t^*}(C) < w_{t^*}(C^*) = 0$. This however would mean that $\lambda(C) - t^*\tau(C) < 0$, or in other words that $W(C) = \frac{\lambda(C)}{\tau(C)} < t^*$, contradicting the assumed minimality of t^*. □

The pseudo-polynomial bound on the execution time quoted in section 4.3 comes about as follows.

Proof. We define λ and τ on sets of edges by summation. Let τ_0 be the maximum value of τ over E. Let C_1 and C_2 be two cycles with distinct ratios. Then

$$\left| \frac{\lambda(C_1)}{\tau(C_1)} - \frac{\lambda(C_2)}{\tau(C_2)} \right| \neq 0$$

or

$$\left|\frac{\lambda(C_1)\tau(C_2) - \lambda(C_2)\tau(C_1)}{\tau(C_1)\tau(C_2)}\right| \neq 0 \tag{14}$$

Since the left hand side of equation 14 is non-zero, and all the data are integer, the numerator must be at least 1 in absolute value. The denominator is at most τ_0^2. Thus on each iteration, the value of t must decrease by at least $\frac{1}{\tau_0^2}$.

Let λ_0 be the maximum absolute value of λ over E. Then again because the data is integral, the minimum ratio weight must lie in the interval $[-\lambda_0, \lambda_0]$. The algorithm therefore cannot iterate more than $2\lambda_0\tau_0^2$ times. Since on each iteration, the negative cycle detection algorithm has time bound $O(mn)$, the pseudo-polynomial bound on the time is $O(\lambda_0\tau_0^2 mn)$. In our case, the edge weights do not depend on the size of the graph, since they are related to the maximum image function value, which is independent of image size. The pseudo-polynomial bound is therefore polynomial in our case. □

The negative cycle detection algorithm used in the minimum ratio weight cycle algorithm was a dequeue implementation of a modified label-correcting algorithm for computing the shortest path lengths from a source vertex s to all $v \in V$. The generic label-correcting algorithm maintains labels d for each vertex. These are upper bounds on the shortest path lengths. It selects edges $e = \langle u, v\rangle$ one at a time and updates them if $d(v) > d(u) + w(e)$, where w is the edge weight function. The modified label-correcting algorithm instead removes vertices u from a list and updates the vertices v for which $\langle u, v\rangle \in E$. If v is not in the list, it is added. Both these algorithms are pseudo-polynomial [1]. There is an $O(mn)$ implementation of the generic label-correcting algorithm that uses a queue as the list structure, adding updated vertices to the back. The dequeue implementation of the modified label-correcting algorithm is pseudo-polynomial but the fastest in practice, especially on sparse graphs such as lattices. In this version, the list is maintained as a dequeue. The vertices are always removed from the front of the queue but may be added to the front or the back. A vertex is added to the front if it has been in the list earlier. Otherwise it is added to the back. The idea is that if v has been seen before, it will have updated some other vertices, its out-neighbours. If it is updated again, it is best to update these other vertices immediately, rather than first remove them from the list with old (and probably out of date) values, and update their out-neighbours, only to have to update those same out-neighbours a second time when v is eventually removed from the list. The time bound on this implementation is $O(\min(nmw_{\max}, m2^n))$, where $w_{\max}$ is the maximum absolute value of w over E. Experiments show that in practice this implementation is approximately linear time.

There are several ways of detecting negative cycles while running these algorithms. If the label of a vertex slips below $-nw_{\max}$, then a negative cycle exists. One can also check whether the predecessor graph from each vertex has a cycle. If it does, this cycle must be negative, since no positive cycle can form part of the predecessor graph. This takes $O(n)$ time, and so does not slow down the algorithm if it is done every αn distance updates.

Finally, when the algorithm terminates, the way to find the zero length cycles is simply to adjust the edge weight for each edge $e = \langle u, v\rangle$ to $\tilde{w}(\langle u, v\rangle) = w(e) + d(v) - d(u)$, where d are the shortest path lengths computed by the label-correcting algorithm. Now a new graph G_0 is formed by removing all edges except those with $\tilde{w}(e) = 0$, along with disconnected vertices. Now cycles in G_0 correspond to zero length cycles in G with edge weights $\tilde{w}$. Finding cycles

in G_0 is accomplished in the standard depth-first labelling fashion, looking for a back edge. In this way, it is possible to find degenerate minima.

There are even more efficient negative cycle detection algorithms based on a transformation to a matching problem [1]. When $w_{\max}$ is polynomial in n, these offer a time-bound of $O(n^{1/2}m\log(nw_{\max}))$. There is also the linear time algorithm for planar graphs mentioned in the text [17].

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: Theory, algorithms and applications*, ch. 5, pp. 133–165, Prentice Hall, NJ, U.S.A., 1993.

[2] A. Amini, S. Tehrani, and T. Weymouth, *Using dynamical programming for minimizing the energy of active contours in the presence of hard constraints*, Proc. Int'l Conf. Comp. Vis., 1988, pp. 95–99.

[3] A. Blake and A. Zisserman, *Visual reconstruction*, MIT Press, Cambridge, MA, U.S.A., 1987.

[4] I. J. Cox, S. B. Rao, and Y. Zhong, *Ratio regions: a technique for image segmentation*, Proc. Int'l Conf. Patt. Rec., vol. 2, 1996, pp. 557–564.

[5] G. B. Dantzig, W. O. Blatner, and M. R. Rao, *Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem*, Theory of Graphs: International Symposium, Gordon and Breach, New York, 1966, pp. 77–84.

[6] J. Elder and S. W. Zucker, *The effect of contour closure on the rapid discrimination of two-dimensional shapes*, Vision Res. **33** (1993), 981–991.

[7] ———, *A measure of closure*, Vision Res. **34** (1994), no. 24, 3361–3369.

[8] ———, *Computing contour closure*, Proc. Euro. Conf. Comp. Vis., June 1996, pp. 399–412.

[9] D. Geiger, A. Gupta, L. Costa, and J. Vlontzos, *Dynamic programming for detecting, tracking, and matching deformable contours*, IEEE Trans. Patt. Anal. Mach. Intell. **17** (1993), no. 3, 294–302.

[10] G. Guy and G. Medioni, *Inferring global perceptual contours from local features*, Int'l J. Comp. Vis. **20** (1996), no. 1-2.

[11] H. Ishikawa and I. H. Jermyn, *Region extraction from multiple images*, Proc. 8$^{\text{th}}$ IEEE Int'l Conf. Comp. Vis. (Vancouver, Canada), 2001.

[12] I. H. Jermyn and H. Ishikawa, *Globally optimal regions and boundaries*, Proc. 7$^{\text{th}}$ IEEE Int'l Conf. Comp. Vis. (Kerkyra, Greece), September 1999, pp. 904–910.

[13] G. Kaniza, *Contours without gradients or cognitive contours*, Italian Jour. Psych. **1** (1971), 93–112.

[14] ———, *Organization in vision: Essays on gestalt perception*, Praeger, New York, U.S.A., 1979.

[15] R. Karp, *A characterization of the minimum cycle mean in a digraph*, Dis. Math. **23** (1978), 309–311.

[16] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active contour models*, Int. J. Comp. Vis. **1** (1988), no. 4, 321–331.

[17] P. Klein, S. Rao, M. Rauch, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, Special issue of J. Comp. and Sys. Sci. **55** (1997), no. 1, 3–23.

[18] I. Kovács and B. Julesz, *A closed curve is much more than an incomplete one: Effect of closure in figure-ground segmentation*, Proc. Natl. Acad. Sci. USA **90** (1993), 7495–7497.

[19] E. L. Lawler, *Optimal cycles in doubly weighted linear graphs*, Theory of Graphs: International Symposium, Gordon and Breach, New York, U.S.A., 1966, pp. 209–213.

[20] T. Leung and J. Malik, *Contour continuity in region based image segmentation*, Proc. Euro. Conf. Comp. Vis. (Germany), 1998.

[21] S. Mahamud, K. K. Thornber, and L. R. Williams, *Segmentation of salient closed contours from real images*, Proc. 7$^{\text{th}}$ IEEE Int'l Conf. Comp. Vis. (Corfu, Greece), 1999.

[22] N. Meggido, *Combinatorial optimization with rational objective functions*, Mathematics of Operations Research **4** (1979), 414–424.

[23] U. Montanari, *On the optimal detection of curves in noisy pictures*, Comm. ACM **15** (1971), no. 5, 335–345.

[24] D. Mumford, *Elastica and computer vision*, Algebraic Geometry and Its Applications (Chandrajit L. Bajaj, ed.), Springer-Verlag, New York, U.S.A., 1994, pp. 491–506.

[25] P. Parent and S. W. Zucker, *Trace inference, curvature consistency, and curve detection*, IEEE Trans. Patt. Anal. Mach. Intell. **11** (1989), no. 8.
[26] A. E. Seiffert and P. Cavanagh, *Position displacement, not velocity, is the cue to motion detection of second-order patterns*, Vision Research **38** (1988), 3569–3582.
[27] A. Sha'ashua and S. Ullman, *Structural saliency: The detection of globally salient structures using a locally connected network*, Proc. Second. Int'l Conf. Comp. Vis. (Florida, U.S.A.), 1988.
[28] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Patt. Anal. Mach. Intell. **22** (2000), no. 8, 888–905.
[29] G. Sperling, *Three stages and two systems of visual processing*, Spatial Vision **4** (1989), no. 2, 183–207.
[30] K. K. Thornber and L. R. Williams, *Analytic solution of stochastic completion fields*, Biological Cybernetics **75** (1996), 141–151.
[31] L. R. Williams and D. W. Jacobs, *Stochastic completion fields: A neural model of contour shape and salience*, Neural Computation **9** (1997), 849–870.
[32] S. C. Zhu and A. Yuille, *Region competition: Unifying snakes, region growing, and Bayes/MDL for multi-band image segmentation*, IEEE Trans. Patt. Anal. Mach. Intell. **18** (1996), no. 9, 884–900.

Ian H. Jermyn, INRIA Sophia Antipolis, 2004 route des Lucioles B.P. 93, 06902 Sophia Antipolis Cedex, France. Email: Ian.Jermyn@sophia.inria.fr Tel: +33 4 92 38 76 83. Fax: +33 4 92 38 76 43.

Hiroshi Ishikawa, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, U.S.A. Tel: +1 212 998 3007. Fax: +1 212 995 4123. Email: ishikawa@cs.nyu.edu. http://cs.nyu.edu/ishikawa/