

Curvature Regularization for Curves and Surfaces in a Global Optimization Framework

Petter Strandmark and Fredrik Kahl

Centre for Mathematical Sciences, Lund University, Sweden
{petter,fredrik}@maths.lth.se

Abstract. Length and area regularization are commonplace for inverse problems today. It has however turned out to be much more difficult to incorporate a curvature prior. In this paper we propose several improvements to a recently proposed framework based on global optimization. We identify and solve an issue with extraneous arcs in the original formulation by introducing region consistency constraints. The mesh geometry is analyzed both from a theoretical and experimental viewpoint and hexagonal meshes are shown to be superior. We demonstrate that adaptively generated meshes significantly improve the performance. Our final contribution is that we generalize the framework to handle mean curvature regularization for 3D surface completion and segmentation.

1 Curvature in Vision

The problem we are interested in solving amounts to minimizing the following energy functional:

$$E(R) = \int_R g(\mathbf{x}) d\mathbf{x} + \int_{\partial R} (\lambda + \gamma\kappa(\mathbf{x})^2) dA(\mathbf{x}), \quad (1)$$

where R is the 2D (or 3D) foreground region with boundary ∂R . Here $g(\mathbf{x})$ is the data term, which may take many forms depending on the application, λ is a positive weighting factor for length (or area) regularization, and γ controls the amount of curvature regularization, denoted κ . Note that the domain may be a 2D image region or a 3D region. In the former case, the boundary is a curve and the notion of curvature is the usual one, while in the latter, the boundary is a surface and κ refers to the mean curvature.

Second order priors like curvature are important for many vision applications, such as stereo [1]. In image segmentation, experiments have shown that curvature regularization is able to capture thin, elongated structures [2,3] where standard length-based regulators would fail. Curvature has also been identified as a key factor in human perception based on psychophysical experiments on contour completion [4] and there is evidence that cells in the visual cortex detect curvature [5]. Still, most segmentation-based approaches in computer vision do not use curvature information. This is contrast to length or area regularity which do play an important role. One of the reasons for this fact is that curvature regularity is

harder to incorporate in a global optimization framework. Note that curvature regularity is fundamentally different from length or area regularity. While, for example, length regularization prefers shorter boundaries, there is no such bias in curvature regularization. In fact, due to a famous theorem of Werner Fenchel, we know that the integral of the absolute curvature for any closed convex plane curve is equal to 2π .

In differential geometry, energy functionals of the type in (1) have been studied for a long time. In the surface case, the functional is known as the Willmore energy [6]. It gives a quantitative measure of how much a given surface deviates from a round sphere. Local descent techniques have been derived for minimizing (1), cf. [7], but they are very dependent on a good initialization.

We propose several improvements to the current state-of-the-art of curvature regularization. This gives both faster running times and smaller memory requirements, and hence important steps are taken to make curvature regularization more practical. More specifically, we (i) solve an identified issue with extraneous arcs while keeping the relaxation tight. We also show how to obtain (ii) better suited tessellations of the domain, and (iii) higher resolution by using adaptive meshes. In turn, this opens up the possibility to apply curvature regularization for surfaces in \mathbf{R}^3 . To our knowledge, this is the first published work that has been able to globally optimize functionals of squared mean curvature.

There are number of application problems that can be modeled by the energy functional in (1). In this paper, we concentrate on improving the methodology for optimizing the functional and we compare with current state-of-the-art methods, cf. [2,3,8]. Experimental results are mainly given for segmentation, but other applications include surface completion [9] and inpainting [3,10].

2 Length and Area Regularization

The basis for our work is the discrete differential geometry framework developed by Sullivan in [11] and Grady in [12] for computing minimal surfaces and shortest paths. The goal is to compute a discrete approximation of the continuous functional in (1). We will recast the problem as an integer linear program and solve it via LP relaxation. In this section we limit the exposition to the standard case without the curvature term (corresponding to $\gamma = 0$). Interestingly, this integer linear program can be shown to be totally unimodular and hence the LP relaxation will be tight.

The method is based on tessellating the domain of interest into a so-called *cell complex*, a collection of non-overlapping basic regions whose union gives the original domain. Several types of tessellations are possible. Some examples are given in Fig. 3 for 2D and Fig. 5 for 3D meshes. Typical choices are square meshes (2D), resulting in 4-connectivity and cube meshes (3D) giving 6-connectivity. To mimic 8-connectivity, pixels are subdivided into four triangular regions each. We will elaborate more on this issue in Section 4.

The boundaries of 2D regions are called *edges*, and the boundaries of 3D regions are called *facets*. To make this approach work, it is necessary to consider both possible orientations of each edge and facet.

In the integer linear program, there are two sets of binary variables, one reflecting regions and the other related to boundaries. For each basic region, a binary variable reflects whether the region belongs to the foreground or the background. Let $x_i, i = 1, \dots, m$ denote these binary variables, where m is the number of basic regions. The region integral in (1) is now easily approximated by a linear objective function of the form $\sum_{i=1}^m g_i x_i$.

In this paper we let x_i denote region variables and y_i boundary variables (lines in 2D and facets in 3D). The length area term in (1) is then represented with $\lambda \sum_i \ell_i y_i$, where ℓ_i denotes the length of edge i , and similarly in 3D with areas a_i . To enforce consistency between the region and boundary variables, **surface continuation constraints** [3] are used in the 2D case. We use completely analogous constraints (8) in three dimensions.

When considering surface completion, region variables are not needed and linear constraints of the type (5) may be used to enforce consistency of the surface [12].

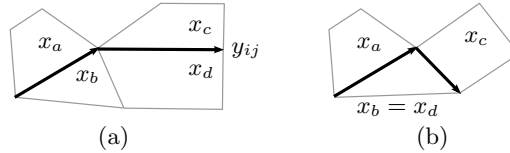


Fig. 1: The line pair variable y_{ij} and its four incident region variables x_a, x_b, x_c and x_d . The four region variables may coincide for some edge pairs.

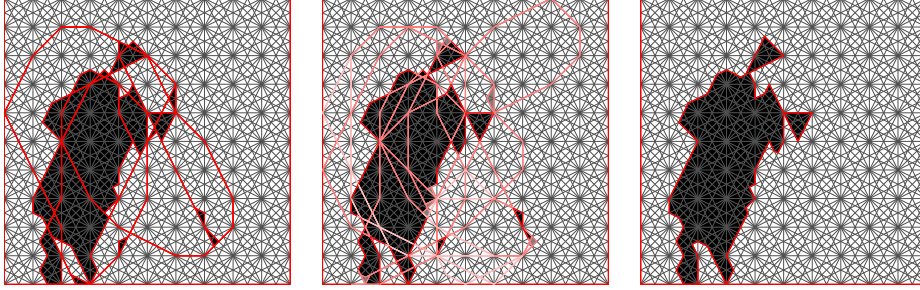
3 Curvature Regularization

To be able to handle curvature regularization, *pairs* of boundary variables are introduced. We denote these pairs by y_{ij} . Schoenemann et al. [3] described how to introduce **boundary continuation constraints** to ensure that an actual boundary curve is formed. Without the latter constraint, only straight line pairs would be used.

Having introduced the line pair variables, the last term in (1) may also be represented as a linear function: $\gamma \sum_{i,j} b_{ij} y_{ij}$. The coefficients b_{ij} used by [13,3] were

$$b_{ij} = \min\{\ell_i, \ell_j\} \left(\frac{\alpha}{\min\{\ell_i, \ell_j\}} \right)^p, \quad (2)$$

where α is the angle difference between the two lines. We use $p = 2$ exclusively in this paper.



(a) Using the original constraints from [3]. The small black regions to the right have their boundary costs reduced by the large arcs of extra boundaries.

(b) Simply requiring that $y_{ij} + y_{ji} \leq 1$ would work if the variables were integers, but causes the LP relaxation to output a fractional solution.

(c) Result using the additional constraints (4). The LP-relaxation output an integral solution.

Fig. 2: Segmentation with and without region consistency constraints. A very crude mesh was used to make the visualization clearer. Gray scale polygons indicate region variables and red lines indicate edge pair variables. Gray lines show the mesh used. Parameters: $\lambda = 0$, $\gamma = 300000$. The time to solve the problem decreased by adding the extra constraints: 0.251s vs. 3.402s for the original problem.

3.1 Avoiding Extraneous Arcs

The constraints introduced by [3] admit too many feasible solutions, which is discussed in the recent work [8]. The problem is illustrated in Fig. 2a, where sharp corners are avoided by introducing large, extra curves which due to the nonexistent length penalty have low cost. This solution is integral and optimal in the original formulation, because along the spurious large arcs both y_{ij} and y_{ji} are active. The result is an underestimation of the boundary curvature, in turn resulting in a non-optimal segmentation. Fig. 2a have several small regions which do not appear in the correct Fig. 2c.

The solution seems simple: to add constraints $y_{ij} + y_{ji} \leq 1$. This would indeed solve the problem if the variables could be restricted to be integral, but we have found the constraints in practice gives a fractional solution with even more spurious arcs (Fig. 2b). Instead, we propose new linear constraints in addition to the constraints in [3]:

Region consistency constraints. Consider a line pair variable y_{ij} and call its four incident regions x_a , x_b , x_c and x_d , located as shown in Fig. 1a. If $x_a = x_b = 1$ or $x_a = x_b = 0$, the region pair should not be active. Similarly for x_c and x_d . This can be linearly encoded as

$$\begin{aligned} x_a + x_b + y_{ij} + y_{ji} &\leq 2 \\ -x_a - x_b + y_{ij} + y_{ji} &\leq 0. \end{aligned} \tag{3}$$

Similar constraints hold for x_c and x_d . All in all, four new constraints are introduced for each pair of edges. It might be the case that x_a and x_c or x_b and x_d coincide, c.f. Fig. 1b. The constraint (3) still looks the same.

To reduce the total number of constraints, we can combine the constraints into four constraints per edge k :

$$\begin{aligned}
 x_{k;1} + x_{k;2} + \sum_{kj \text{ a pair}} y_{kj} &\leq 2 \\
 -x_{k;1} - x_{k;2} + \sum_{kj \text{ a pair}} y_{kj} &\leq 0 \\
 x_{k;1} + x_{k;2} + \sum_{jk \text{ a pair}} y_{jk} &\leq 2 \\
 -x_{k;1} - x_{k;2} + \sum_{jk \text{ a pair}} y_{jk} &\leq 0.
 \end{aligned} \tag{4}$$

Here $x_{k;1}$ and $x_{k;2}$ denote the two regions adjacent to edge k . The first two constraints sum over all line pairs starting with edge k and the last two sum over all pairs ending with edge k .

Fig. 2c show the result with these additional constraints where the boundary now is consistent with the region variables. As a bonus, the new constraints reduced the time required to solve the problem to about 7%. We can also see from Fig. 2 that both before and after the additional constraints, the optimal solution has its region variables equal or very close to 0 or 1.

3.2 Curvature of Surfaces

Each facet in our 3D mesh is associated with a variable $\mathbf{y} = (y_1, \dots, y_{2n})$ of areas $\mathbf{a} = (a_1, \dots, a_{2n})$. There are twice as many variables as facets, because each facet is associated with two variables, one for each orientation. The two are distinguished by (arbitrarily) assigning a normal to each face in the mesh. With the matrix \mathbf{B} as defined in [12], the optimization problem for surface completion with area regularization is

$$\begin{aligned}
 &\underset{\mathbf{y}}{\text{minimize}} \quad \lambda \mathbf{a}^T \mathbf{y} \\
 &\text{subject to} \quad \mathbf{B}\mathbf{y} = 0; \quad \mathbf{y} \in \{0, 1\}^{2n}; \quad y_k = 1, k \in K.
 \end{aligned} \tag{5}$$

$$\mathbf{B}_{e,y_i} = \begin{cases} +1, & \text{if edge } e \text{ borders } y_i \text{ with coherent orientation} \\ -1, & \text{if edge } e \text{ borders } y_i \text{ with incoherent orientation} \\ 0, & \text{otherwise} \end{cases}$$

K is the set of facets that are supposed to be part of the minimal surface a priori. We now extend this formulation to support curvature by introducing face pairs. Each pair of facets in the mesh with an edge in common are associated with two variables $\{y_{ij}\}$ (one for each orientation). Enforcing consistency between the face variables and the variables corresponding to the pairs of faces can be done with linear constraints:

Surface continuation constraints. For each oriented facet k and each one of its edges e we add the following constraint:

$$y_k = \sum_{ij \text{ with edge } e} d_{k,ij} y_{ij}. \quad (6)$$

The sum is over all pairs ij with edge e in common. The indicator $d_{k,ij}$ is 1 if facet k is part of the pair ij .

Having introduced the facet pairs, we follow [14] and associate them with a cost $b_{i,j}$, approximating the mean curvature (compare with (2) on page 3):

$$b_{i,j} = \frac{3\|\mathbf{e}_{i,j}\|^2}{2(a_i + a_j)} \left(2 \cos \frac{\theta_{i,j}}{2} \right)^2, \quad (7)$$

where $\theta_{i,j}$ is the dihedral angle between the two facets in the pair. $\|\mathbf{e}_{i,j}\|$ is the length of their common edge. The objective function we are minimizing is then $\rho \sum_i a_i y_i + \sigma \sum_{i,j} b_{i,j} y_{i,j}$, subject to the constraints in (5) and (6). This approximation is not perfect; for example, it will not give the correct approximation for saddle points. However, it measures how much the surface bends and fulfills a couple of requirements listed by [14].

Segmentation, as opposed to surface completion, requires variables for each volume element in order to incorporate the data term. Additional consistency constraints are then required:

Volume continuation constraints. For each facet k ,

$$\sum_i b_{k,i} y_i + \sum_i g_{k,i} x_i = 0, \quad (8)$$

where b_k indicates whether the facet y_k is positively or negatively incident w.r.t. the chosen face normal. $g_{k,i}$ is 1 if the volume element x_i is positive incident (the face normal points towards its center), -1 if it is negative incident and 0 otherwise. Both sums have two non-zero terms.

3.3 Pseudo-Boolean Optimization

Solving the discrete optimization problem does not have to be done using a linear program. It is also possible to use discrete optimization methods such as roof duality [15]. Each edge pair is represented as a 3- or 4-clique in the energy minimization. This formulation has the advantage that it readily carries over to three dimensions. El-Zehiry and Grady [2] used 3-cliques for minimizing curvature functionals and their formulation is equivalent to [3] for 4-connected grids. This is because in a 4-connected grid, only configurations of the type in Fig. 1b are present. For all connectivities higher than 4, many edge pairs are adjacent to 4 regions, see Fig. 1a. The cost of an edge pair can be written as:

$$b_{i,j} \left(x_a x_c (1 - x_b) (1 - x_d) + (1 - x_a) (1 - x_c) x_b x_d \right). \quad (9)$$

Representing this, however, requires extra nodes to be added, since the degree-4 term does not disappear.

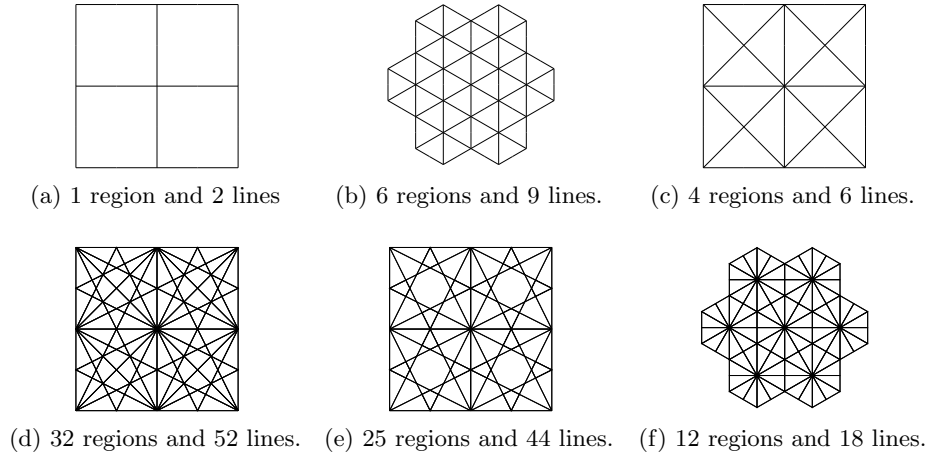


Fig. 3: Different types of grids with the number of regions and average number of lines per cell. The maximum angle between the possible straight lines is 90° in (a), 60° in (b) and 45° in (c). Meshes (d), (e) and (f) have about 27° , 37° and 30° as their maximum angle, respectively.

4 Tessellations

The mesh used for the segmentation can be created in a number of ways. The quality of the approximation depends on how many different possible straight lines that can be represented by the mesh, since a larger possible choice of line slopes allows the mesh to approximate a continuous curve more closely. Fig. 3 shows some possible meshes and the straight lines they admit. If a mesh allows n possible straight line directions, it is referred to as n -connected.

4.1 Hexagonal Meshes

Hexagonal meshes have long been studied for image processing [16]. One characterizing fact of hexagons is that they are the optimal way of subdividing a surface into regions of equal area while minimizing the sum of the boundary lengths [17]. The fact that is more important to us is the neighborhood structure. In a hexagonal lattice every region has 6 equidistant neighbors. When approximating curvature we would like to represent as many different straight lines as possible and we would like the maximum angle between them to be small, as that gives us a better approximation of a smooth curve [13]. The neighborhood structure of the hexagonal mesh allows for similar performance (number of lines and angle between them) while using fewer regions. This is illustrated in Fig. 3, where three crude meshes and three finer meshes are shown. The meshes in Figs. 3d and 3f have similar maximal angle between the possible straight lines, but the hexagonal mesh achieves this with fewer regions due to the favorable

intersection pattern of the lines. This suggests that hexagonal meshes can achieve the same accuracy as the meshes (c) and (d) used in [3] with a significantly smaller linear program.

Calculating the data term. With the introduction of the hexagonal grid, every region is no longer contained within a single pixel. Some regions will partly overlap more than one pixel. The data term for the region R_k is the integral of g over that region:

$$g_k = \int_{R_k} g(\mathbf{x}) d\mathbf{x}. \quad (10)$$

The data term is a function on a continuous domain. However, because it arises from a measured image, it will be piecewise constant on the $n \times m$ square image pixels. If each pixel is subdivided into regions, the data term for each region is simply the area of the region multiplied with the data term for the pixel. In the general case, the data term for R_k is computed as:

$$g_k = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} g\left(i + \frac{1}{2}, j + \frac{1}{2}\right) \cdot \text{area}(R_k \cap p_{ij}), \quad (11)$$

where p_{ij} is the square representing pixel (i, j) :

$$p_{ij} = \{(x, y) \mid i \leq x \leq i + 1, j \leq y \leq j + 1\}. \quad (12)$$

Calculating this sum requires a large number of polygon intersections to be computed. For this we used the General Polygon Clipper library (GPC) from the The University of Manchester.

4.2 Adaptive Meshes

The memory requirements for solving the linear programs arising from the discretizations are very large. Each pair of connected edges introduce two variables. Linear programs are typically solved using the simplex method or interior point methods, both of which require a substantial amount of memory for our problems. As one example, a problem with 131,072 regions and 1,173,136 edge pairs required about 2.5 GB of memory to solve using the Clp solver.

For this reason, it is desirable to keep the size of the mesh small. However, a fine mesh is needed to be able to approximate every possible curve. The solution to this conflict of interest is to generate the mesh adaptively, to only give it high resolution where the segmentation boundary is likely to pass through. Adaptive meshes have previously been considered for image segmentation in the level-set framework [18] and in combinatorial optimization of continuous functionals [19].

The mesh is refined using an iterative process. First, a single region is put into a priority queue. Then regions are removed from the priority queue and subdivided into smaller regions which are put back into the queue. The region which most urgently needs to be split is removed first from the priority queue.

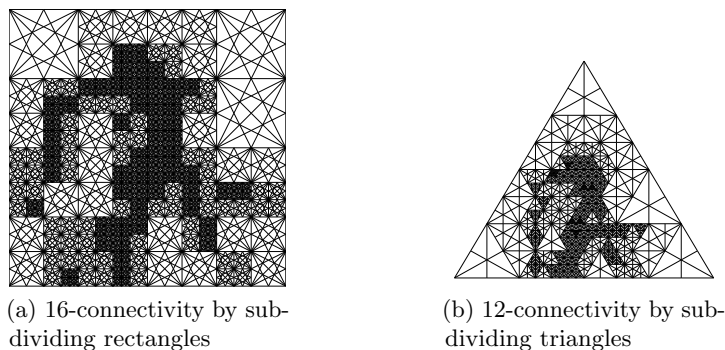


Fig. 4: Adaptive meshes can be constructed by recursively subdividing basic shapes into several similar shapes and finally adding the extra connectivity.

```

Start with  $q$  an empty priority queue
 $R \leftarrow (0, 0, w, h)$ 
Add  $R$  to  $q$  with priority =  $\text{score}(R)$ 
while  $\text{size}(q) < L$  do
  Remove  $R$  from  $q$ 
  Split  $R$  into  $R_1 \dots R_k$ 
  Add  $R_1 \dots R_k$  to  $q$  with  $\text{score}(R_1) \dots \text{score}(R_k)$ 
end while

```

Both square and triangular basic shapes can be split up into four identical shapes similar to the original one. Thus, for all adaptive meshes $k = 4$. The score function can be chosen in many different ways. One way is to use the squared deviation from the mean of each region, i.e.:

$$\text{score}(R) = \int_R (I(\mathbf{x}) - \mu(R))^2 d\mathbf{x}, \quad (13)$$

where $\mu(R) = \frac{1}{|R|} \int_R I(\mathbf{x}) d\mathbf{x}$. This way, regions where the data term vary a lot will be split before regions which have a uniform data term. The score is not normalized, because otherwise many very small regions would tend to have a big score. The integrals may be computed in the same manner as (11) for images with square pixels, resulting in the computation of polygon intersections.

4.3 Tetrahedrons

There are many choices of tessellations in three dimensions. We have taken a quite simple approach and divided each unit cube into five tetrahedrons, as shown in Fig. 5. This allowed us to be enough different planes to demonstrate the global optimization of mean curvature.

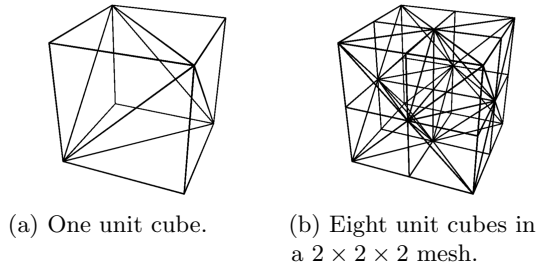


Fig. 5: *Interactive figure*. Each unit cube is split into 5 tetrahedrons. This is the type of mesh used for our experiments in 3D. When stacking several, every other cube has to be mirrored in order to fit.

5 Experimental Results

This paper does not focus on how to model the data term and we will use a simple, two-phase version throughout all our experiments:

$$g(\mathbf{x}) = (I(x) - \mu_1)^2 - (I(x) - \mu_0)^2, \quad (14)$$

where μ_0 and μ_1 are two fixed mean values and I is the image.

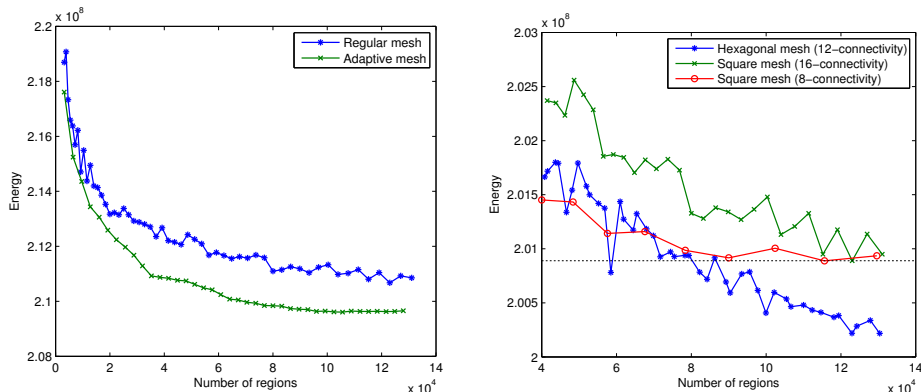
5.1 Hexagonal Meshes

In our first experiment we evaluate hexagonal vs. square meshes. We are comparing three types of meshes, the 8- and 16-connected square mesh and the 12-connected hexagonal mesh, shown in Fig. 3 (c), (d) and (f). We fixed a data term of a 256×256 image (cameraman) and lay meshes of various types and sizes on top of it and calculated the optimal energy.

The result is shown in Fig. 6b, where the optimal energy is plotted as a function of the number of regions used. This is reasonable, since the number of regions is a good indicator of the total size of the linear program. The analogous plots using the number of line pairs or edges look the same. We see that the 8-connected grid converges quickly, but to a suboptimal energy. The hexagonal mesh consistently outperforms the 16-connected grid. If we were to let the number of regions grow very large, the 16-connected grid would probably achieve a lower energy than the hexagonal, due to it having 2 more possible straight lines. We have not been able to observe this in practice, though, due to the memory requirements.

5.2 Adaptive Meshes

To evaluate the effect of adaptive meshes, we performed a number of experiments. Firstly, we evaluated the visual quality of the segmentation for regular and



(a) Optimal energy vs. the total number of regions for a square 16-connected mesh. To get the same accuracy as the finest parts of the adaptive mesh, the regular mesh would need $210 \cdot 10^4$ regions. In contrast, the adaptive mesh converged using about $10 \cdot 10^4$ regions.

(b) Optimal energy vs. the total number of regions. The best accuracy obtained by the square mesh was achieved by the hexagonal mesh with about half the number of regions.

Fig. 6: Experiments evaluating adaptive and hexagonal meshes. These experiments used $\lambda = \gamma = 10000$. The energy difference might seem small, but differences of these magnitudes often correspond to significant changes in segmentation, cf. Fig. 7.

adaptive 16-connected meshes with the same number of regions. The result can be seen in Fig. 7. There is a significant visual difference. The fact that the adaptive mesh achieved a smoother curve is also reflected in the optimal energy, which is lower. The results for 8-connected meshes are shown in Fig. 7d and yield the same conclusion.

To evaluate the performance more quantitatively, we solve the same segmentation problem a large number of times for different number of regions. The adaptive mesh converged to what probably is the optimal energy for that connectivity, while the regular mesh did not. The regular mesh would have required more than 20 times more regions to achieve the same energy. Fig. 6a shows the optimal energies for the different number of regions and the two types of meshes.

5.3 The Wilmore Functional

For our experiments in three dimensions we generated a mesh where each unit cube was split into 5 tetrahedrons, see Fig. 5. We then created the set K as two circular surfaces at $z = 0$ and $z = z_{\max}$, with nothing in between. The analytic solution with area penalty is the catenoid, one of the first minimal surfaces found. Fig. 8a show the discrete version obtained with $\lambda = 1$ and $\gamma = 0$. If instead the mean curvature is chosen as the regularizer, the optimal surface instead bends outwards. The solution to this problem is shown in Fig. 8b and is the global optimum, since all variables ended up integral in the LP relaxation of (5). We used Clp as our LP solver.

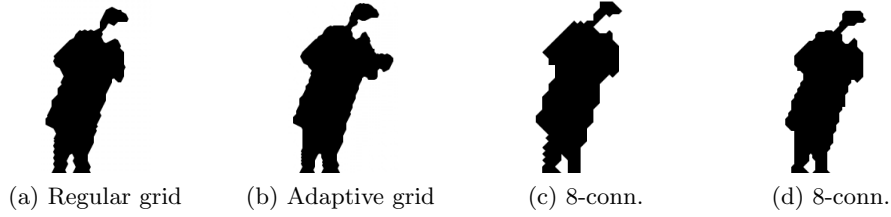


Fig. 7: Results with (a) regular and (b) adaptive grids. The number of regions used were 32,768 in both cases and the number of edge pairs were 291,664 and 285,056, respectively. The adaptive mesh gives a smoother curve and correctly includes the hand of the camera man. The optimal energy for the regular mesh was $2.470 \cdot 10^8$ and $2.458 \cdot 10^8$ for the adaptive. This experiment used $\lambda = 30000$ and $\gamma = 1000$. (a) and (b) used 16-connectivity (Fig. 3d), whereas (c) and (d) show the same experiment with 8-connectivity (Fig. 3c).

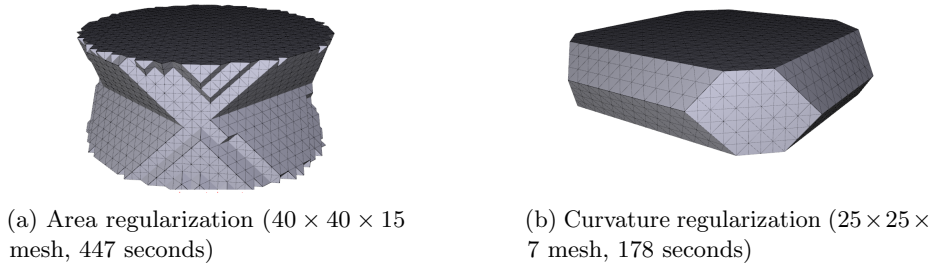


Fig. 8: *Interactive figure.* Surface completion with area and curvature regularization. Two flat, circular surfaces at the top and bottom were fixed. The surface in (a) bends inwards to approximate a catenoid and in (b) it correctly bends outwards to minimize the squared mean curvature.

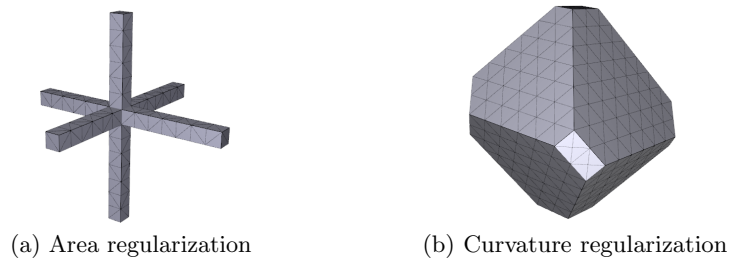


Fig. 9: *Interactive figure.* Surface completion on a $16 \times 16 \times 16$ mesh with area and curvature regularization and volume element variables. The data term and the optimal surface using area regularization coincide. The radius of the volume in (b) is constrained by the mesh size. Otherwise, a minimal surface would not exist for the continuous problem.

In another experiment we also used variables for the volume elements. The data term was a 3D ‘cross’ where the volume elements were forced to be equal to 1, whereas the volume elements at the boundary were forced to be 0. The optimal segmentation when the area was minimized coincided with the data term and is shown in Fig. 9a. When instead minimizing the curvature the optimal segmentation should resemble a sphere, which is observed in Fig. 9b.

5.4 Pseudo-Boolean Optimization

Finally, we have compared linear programming to the pseudo-Boolean formulation from Section 3.3. It seems that the formulation with higher-order cliques is weaker than the previously discussed linear programming formulations. In all cases except with a negligible value of γ we obtained 75%–100% unlabeled nodes, with no hope of recovering a good solution with e.g. probing [15].

6 Conclusions

The purpose of this paper has been to discuss the problem of segmentation with curvature regularization and to enhance the methodology in numerous ways.

First of all, we have introduced new region consistency constraints (4) which are essential for the method in [3] to work well (Fig. 2c). These new constraints also reduced the computation time to less than one tenth of the original time.

We have argued, by regarding the angles between straight lines and the number of regions in different meshes that hexagonal meshes are more suitable than square meshes with the same number of regions. Our experiments have confirmed this conclusion as well (Fig. 6b).

Another way of reducing the memory requirements is to allow for an adaptive mesh. We have shown that generating the mesh adaptively by examining the changes of the data term results in far better segmentations, both quantitatively (Fig. 6a) and qualitatively (Figs. 7 and 7d).

Lastly, we have introduced constraints for 3D surface completion and segmentation. Experiments are encouraging (Figs. 8 and 9) with exclusively globally optimal solutions. To our knowledge, this is the first time the mean curvature of surfaces has been optimized globally. The next step would be to apply this method to e.g. the partial surfaces obtained by stereo estimation algorithms. Another line of further research is how to be able to cope with a finer discretization of the 3D volume.

Source code. To facilitate further research, the source code used for the experiments in this paper will be made publicly available and may be downloaded from our web page.

Acknowledgments

We thank Thomas Schoenemann for helpful discussions and sharing his source code. This work has been funded by the Swedish Foundation for Strategic Research

(SSF) through the programmes *Future Research Leaders* and *Wearable Visual Information Systems* and by the European Research Council (GlobalVision grant no. 209480).

References

1. Woodford, O., Torr, P., Reid, I., Fitzgibbon, A.: Global stereo reconstruction under second order smoothness priors. *IEEE Trans. Pattern Analysis and Machine Intelligence* **31** (2009) 2115–2128 [1](#)
2. El-Zehiry, N., Grady, L.: Fast global optimization of curvature. In: *Conf. Computer Vision and Pattern Recognition*. (2010) [1](#), [2](#), [6](#)
3. Schoenemann, T., Kahl, F., Cremers, D.: Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation. In: *Int. Conf. Computer Vision*. (2009) [1](#), [2](#), [3](#), [4](#), [6](#), [8](#), [13](#)
4. Kanizsa, G.: Contours without gradients or cognitive contours. *Italian Jour. Psych.* **1** (1971) 93–112 [1](#)
5. Dobbins, A., Zucker, S.W., Cynader, M.S.: Endstopped neurons in the visual cortex as a substrate for calculating curvature. *Nature* **329** (1987) 438–441 [1](#)
6. Willmore, T.: Note on embedded surfaces. *An. Sti. Univ. "Al. I. Cuza" Iasi Sect. I a Mat. (N.S.)* (1965) 493–496 [2](#)
7. Hsu, L., Kusner, R., Sullivan, J.: Minimizing the squared mean curvature integral for surfaces in space forms. *Experimental Mathematics* **1** (1992) 191–207 [2](#)
8. Schoenemann, T., Kuang, Y., Kahl, F.: Curvature regularity for multi-label problems — standard and customized linear programming. In: *EMMCVPR. Lecture Notes in Computer Science*, Springer (2011) [2](#), [4](#)
9. Kawai, N., Sato, T., Yokoya, N.: Efficient surface completion using principal curvature and its evaluation. In: *Int. Conf. Image Processing*. (2009) 521–524 [2](#)
10. Masnou, S.: Disocclusion: A variational approach using level lines. *IEEE Transactions on Image Processing* **11** (2002) 68–76 [2](#)
11. Sullivan, J.: *Crystalline Approximation Theorem for Hypersurfaces*. PhD thesis, Princeton Univ. (1990) [2](#)
12. Grady, L.: Minimal surfaces extend shortest path segmentation methods to 3D. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **32** (2010) 321–334 [2](#), [3](#), [5](#)
13. Bruckstein, A.M., Netravali, A.N., Richardson, T.J.: Epi-convergence of discrete elastica. *Applicable Analysis, Bob Carroll Special Issue* **79** (2001) 137–171 [3](#), [7](#)
14. Wardetzky, M., Bergou, M., Harmon, D., Zorin, D., Grinspun, E.: Discrete quadratic curvature energies. *Comput. Aided Geom. Des.* **24** (2007) 499–518 [6](#)
15. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: *Conf. Computer Vision and Pattern Recognition*. (2007) [6](#), [13](#)
16. Middleton, L., Sivaswamy, J.: *Hexagonal Image Processing: A Practical Approach*. Springer-Verlag New York, Inc. (2005) [7](#)
17. Hales, T.C.: The honeycomb conjecture. *Discrete & Computational Geometry* **25** (2001) 1–22 [7](#)
18. Xu, M., Thompson, P.M., Toga, A.W.: An adaptive level set segmentation on a triangulated mesh. *IEEE Trans. on Medical Imaging* **23** (2004) 191–201 [8](#)
19. Kirsanov, D., Gortler, S.J.: A discrete global minimization algorithm for continuous variational problems. Technical Report TR-14-04, Harvard (2004) [8](#)