A Robust Level Set Algorithm for Image Segmentation and its Parallel Implementation

Joris R. Rommelse*
Department of Applied Mathematics
Delft University of Technology
Delft, 2628 CD, The Netherlands
Email: J.R.Rommelse@its.tudelft.nl

Hai-Xiang Lin
Department of Applied Mathematics
Delft University of Technology
Delft, 2628 CD, The Netherlands
Email: H.X.Lin@its.tudelft.nl

Tony F. Chan
Department of Mathematics
University of California, Los Angeles
Los Angeles, CA 90095-1555, USA
Email: Chan@math.ucla.edu

ABSTRACT

In this paper we discuss a classic clustering algorithm that can be used to segment images and a recently developed active contour image segmentation model. We propose integrating aspects of the classic algorithm to improve the active contour model. For the resulting CVK segmentation algorithm we examine two methods to decrease the size of the image domain.
The CVK method has been implemented to run on parallel and distributed computers. By changing the order of updating the pixels, it was possible to replace synchronous communication with asynchronous communication and subsequently the parallel efficiency is improved.

Keywords: Image Segmentation, Clustering Algorithm, Active Contour Model, Level Set Functions, Synchronous and Asynchronous Communication.

1. INTRODUCTION

Looking at an image, it is usually very easy for a human to see what it represents. For a computer this is not so easy. Before computers "know" what is represented, objects must be measured, but before that, the objects must first be detected. Understanding images is very important in problems like stereo and motion estimation, part recognition or image indexing. The first step in image understanding is image segmentation. Segmentation is the problem of dividing an image into objects or distinguishing objects from a background.

This paper discusses the classic K-means algorithm, the recently developed Chan - Vese (CV) active contour model and a combination of both (CVK). K-means can only handle a small subset of images and needs image enhancement for noisy or blurred images. CV and CVK are designed to handle a much larger class of images without enhancement. Unfortunately these algorithms are often slower. Without any optimizations, a typical 600×480 image will take several hours. Therefore methods that reduce the size of the image domain and parallel and distributed computers are used to speedup calculation.

Section 1 introduces image segmentation and the K-means, CV and CVK algorithms. In section 2 a narrow band version and a multiresolution version of the CVK algorithm are examined to decrease the size of the image domain. Section 3 discusses the parallel implementation of these versions and shows experiments with synchronous and asynchronous inter-processor communication. Section 4 gives some general conclusions.

Image Segmentation

For every point in an image domain, $\mathbf{x} \in \Omega$, the intensity of a grey-valued image can be specified by a number in [0,1] (0 for black, 1 for white) and the intensity of a RGB image (red, green, blue) can be specified by three numbers in [0,1]. In general, the intensity of images considered in this paper can be specified by m numbers in [0,1]. Therefore images are mappings from Ω to $[0,1]^m$ and can be written as

$$\mathbf{u}:\Omega \to [0,1]^m \quad , \quad \mathbf{u}(\mathbf{x}) = (u_0(\mathbf{x}),\cdots,u_{m-1}(\mathbf{x})) \quad , \quad \mathbf{x}\in\Omega \qquad (1)$$

The image domain can be discrete (a grid of points or pixels) or continuous, has a rectangular shape and can have any number of dimensions larger than one.
For example,

$$\Omega = [0, w_0]\times[0, w_1] \qquad (2)$$

for a 2D continuous image, or

* Supported in part by Universiteitsfonds Delft

$$\Omega = \{0,\cdots,\omega_0 - 1\} \times \cdots \times \{0,\cdots,\omega_{d-1} - 1\} \quad (3)$$

for a d-dimensional digital image. Here w_0 and w_1 are the width and height of the image and ω_i is the number of grid points in the (i+1)-th dimension.

The goal of image segmentation is to segment the image domain Ω into several subdomains Ω_i, based on some appropriate criteria that involve intensity or location of colors, such that the domain is formed by the union of the subdomains and the subdomains do not overlap,

$$\Omega = \bigcup_{i=0}^{k-1} \Omega_i \quad , \quad \Omega_i \cap \Omega_j = \varnothing \quad (i \neq j)\,. \quad (4)$$

After segmentation is completed, the subdomains are considered k separate objects, or k-1 separate objects and a background. The objects can then be measured and classified or recognized. This step towards image understanding is not covered by this paper.

Some Commonly Used Segmentation Algorithms

K-means clustering algorithm [6]

The K-means clustering algorithm was designed to cluster a number (N) of objects into K clusters or classes, based on location of the objects. The objective is to assign the objects to classes so that they lie closer to the average location of their class than to the average location of other classes.

The algorithm can be used to cluster pixels of a digital image with related color if the pixels are considered objects. Location is measured in terms of color rather than the actual position of the pixels in the image; pixels are located in color space, not in physical space. The average location in color space of pixels in a class can be calculated by averaging the colors of all pixels in the class. The distance between a pixel p_i and the average of a class a_j, can be expressed by

$$d(p_i, a_j) = \tfrac{1}{m} \sum_{k=0}^{m-1} \lambda_{j,k} (p_{i,k} - a_{j,k})^2 \,, \quad (5)$$

(m=1 for a grey valued image and m=3 for a RGB image). The parameters $\lambda_{j,k}$ can be used to give priority to classes or colors.

Chan - Vese image segmentation [4,10,11]

A very different approach is segmentation by curve evolution, snakes or active contour models. Here a parameterized hypersurface C (or contour in 2D) moves through the image domain with respect to constraints from the image and stops on the boundaries of objects. Mumford & Shaw base the constraints on the minimization of an energy functional

$$F^{MS}(\bar{\mathbf{u}}, C) = \mu \cdot surface(C) + \lambda \int_{\Omega} |\mathbf{u} - \bar{\mathbf{u}}|^2 \, d\mathbf{x} + \int_{\Omega \backslash C} |\nabla \bar{\mathbf{u}}|^2 \, d\mathbf{x} \quad (6)$$

where $\mathbf{u}$ is the original image, $\bar{\mathbf{u}}$ is the segmented image, C is a hypersurface around detected objects, *surface*(C) is the length of the contour (in 2D) or the area of the surface or hypersurface (in 3D or higher dimensions), $\mu \geq 0$ and $\lambda \geq 0$ are parameters that can be set by the user.

Chan & Vese represent the hypersurface C by a set of n simple closed hypersurfaces $\{C_0, \cdots, C_{n-1}\}$,

$$C = \bigcup_{i=0}^{n-1} C_i \,. \quad (7)$$

Every hypersurface C_i segments $\Omega \backslash C$ in 2 subdomains and the set of n hypersurfaces therefore segments $\Omega \backslash C$ in 2^n subdomains $\{\Omega_0, \cdots, \Omega_{2^n - 1}\}$.

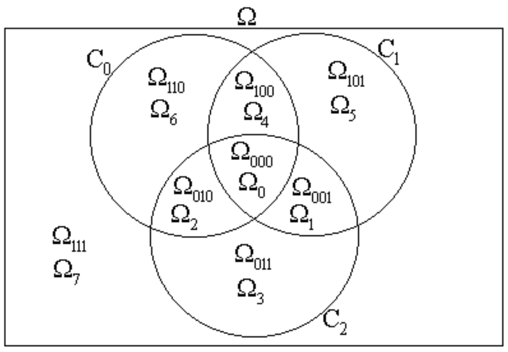

Figure 1. Segmentation of the image domain Ω in eight subdomains $\Omega_0,\ldots,\Omega_7$ by three hypersurfaces C_0, C_1, C_2. The binary representation of the index i of subdomain Ω_i can be found by the relative position to the hypersurfaces. Example: Ω_{001} lies outside (1) C_0, inside (0) C_1 and inside (0) C_2.

Furthermore, they restrict $\bar{\mathbf{u}}$ to piecewise constant functions; $\bar{\mathbf{u}}$ has a constant value $\mathbf{c}_i$ on every subdomain Ω_i

$$\begin{aligned} \bar{\mathbf{u}}(\mathbf{x}) &= \mathbf{c}_i \text{ if } \mathbf{x} \in \Omega_i \\ \bar{\mathbf{u}}(\mathbf{x}) &= \sum_{i=0}^{2^n - 1} \mathbf{c}_i \chi_i(\mathbf{x}) \end{aligned} \quad (8)$$

with

$$\chi_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_i \\ 0 & \text{if } \mathbf{x} \notin \Omega_i \end{cases} \quad (9)$$

the characteristic function of subdomain Ω_i.

Now the Mumford-Shaw energy functional Eq. (6) becomes

$$\begin{aligned} F_n^{MS}(C_0, \cdots, C_{n-1}, \mathbf{c}_0, \cdots, \mathbf{c}_{2^n - 1}) &= \\ = \mu \cdot surface\left(\bigcup_{i=0}^{n-1} C_i \right) + \sum_{i=0}^{2^n - 1} \lambda_i \int_{\Omega_i} |\mathbf{u} - \mathbf{c}_i|^2 \, d\mathbf{x} \end{aligned} \quad (10)$$

For vector-valued images (m>1), the Chan-Vese energy functional is defined by

$$\begin{aligned} F_{n,m}^{CV}(C_0, \cdots, C_{n-1}, \mathbf{c}_0, \cdots, \mathbf{c}_{2^n - 1}) &= \\ = \mu \cdot \sum_{i=0}^{n-1} surface(C_i) + \tfrac{1}{m} \sum_{i=0}^{2^n - 1} \sum_{j=0}^{m-1} \lambda_i \int_{\Omega_i} (u_j - c_{i,j})^2 \chi_i d\mathbf{x} \end{aligned} \quad (11)$$

The steps of the algorithm are to minimize the energy by moving the hypersurfaces while keeping $\{\mathbf{c}_0, \cdots, \mathbf{c}_{2^n - 1}\}$ fixed, and then recalculate $\{\mathbf{c}_0, \cdots, \mathbf{c}_{2^n - 1}\}$ while keeping the hypersurfaces fixed. The latter can be done by setting the derivatives with respect to $c_{i,j}$ to zero:

$$\frac{\partial F_{n,m}^{CV}}{\partial c_{i,j}} = 0 \quad \Rightarrow \quad c_{i,j} = \frac{\int_{\Omega} u_j \chi_i d\mathbf{x}}{\int_{\Omega} \chi_i d\mathbf{x}} = \frac{\int_{\Omega_i} u_j d\mathbf{x}}{\int_{\Omega_i} d\mathbf{x}} = \frac{\int_{\Omega_i} u_j d\mathbf{x}}{|\Omega_i|} \tag{12}$$

In other words, $\mathbf{c}_i$ is the average color of the image on subdomain Ω_i.

For the former, an appropriate representation for the hypersurfaces must be chosen. Chan & Vese choose to use level set functions. On a d-dimensional domain Ω, a hypersurface C is a (d-1)-dimensional object, but it can be represented by a function on Ω; $\phi : \Omega \to \mathbb{R}$.

$$C = \{\mathbf{x} \in \Omega : \phi(\mathbf{x}) = 0\} \tag{13}$$

C is the zero level set of ϕ and $\mathbf{n}(\mathbf{x}_0) = \frac{\nabla \phi}{|\nabla \phi|}$ is a vector normal to $\{\mathbf{x} \in \Omega : \phi(\mathbf{x}) = \phi(\mathbf{x}_0)\}$ at position $\mathbf{x}_0$ pointing in the direction of level sets with $\phi(\mathbf{x}) > \phi(\mathbf{x}_0)$.

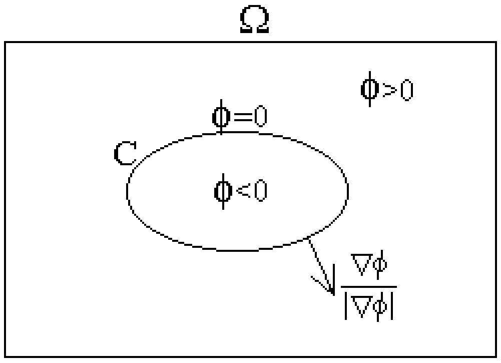

Figure 2. Representation of a hypersurface C and the unit normal by a level set function ϕ on Ω.

Moreover, the position of a moving hypersurface can be calculated by solving an evolution equation for ϕ:

$$\phi_t + v|\nabla \phi| = 0 \quad , \quad \text{given } \phi(\mathbf{x},t) \tag{14}$$

Not just the zero level set, but all the level sets move in normal direction with speed $v(\mathbf{x},t)$.

To segment the image, a suitable speed function $v(\mathbf{x},t)$ can be derived for every hypersurface C_i:

$$\frac{\partial \phi_i}{\partial t} = |\nabla \phi_i| \left(\mu \nabla \cdot \left(\frac{\nabla \phi_i}{|\nabla \phi_i|} \right) - sign(\phi_i)(p_I - p_{J_i}) \right) \tag{15}$$

Here $I(\mathbf{x})$ is the index of the subdomain where $\mathbf{x}$ lies, $J_i(\mathbf{x})$ is the index of the subdomain that lies opposite subdomain $I(\mathbf{x})$ relative to hypersurface C_i and $p_i : \Omega \to \mathbb{R}$ are penalty functions to express why points in Ω should not be in Ω_i,

$$p_i(\mathbf{x}) = \tfrac{1}{m} \sum_{j=0}^{m-1} \lambda_{i,j} (u_j - c_{i,j})^2 \ . \tag{16}$$

Notice that these penalties are also used in the K-means algorithm.

In digital image processing, the PDE Eq. (15) is discretized using central differences for the spatial derivatives and Euler forward for the time derivatives, to fit the given grid. This means that for stability reasons, the time step must be chosen depending on the given spatial step.

CVK algorithm [8]

In the CV algorithm, the color of a pixel is compared to the mean color of its subdomain and the mean colors of the subdomains that lie opposite the hypersurfaces. Therefore a pixel can stay in its subdomain or move to one of n other subdomains (if there are n hypersurfaces). However, the K-means algorithm allows pixels to move to any of the other $2n$-1 subdomain. Apparently in the CV algorithm, pixels might be denied the opportunity to move to the right subdomain.

The CVK algorithm segments an image by evolving hypersurfaces according to the PDE

$$\frac{\partial \phi_i(\mathbf{x})}{\partial t} = |\nabla \phi_i(\mathbf{x})| \left(\mu \nabla \cdot \left(\frac{\nabla \phi_i(\mathbf{x})}{|\nabla \phi_i(\mathbf{x})|} \right) - (1-\mu) sign(\phi_i(\mathbf{y})) \right) \tag{17}$$

where $\mu \in [0,1]$ and

$$\mathbf{y} = \operatorname*{arg\,min}_{\mathbf{z} \in \Omega} \left\{ \tfrac{1}{m} \sum_{j=0}^{m-1} \lambda_{i,j} (u_j(\mathbf{z}) - c_{i,j}(\mathbf{z}))^2 \right\} \tag{18}$$

is a pixel in the subdomain where the penalty function Eq. (16) is minimized, so where a pixel $\mathbf{x}$ should be moved to according to the K-means algorithm. Although there are many such pixels $\mathbf{y}$, the level set function ϕ_i has the same sign on all of them. The sign function in Eq. (17) makes sure that the level set function ϕ_i on pixel $\mathbf{x}$ gets closer to zero or even changes sign when $\mathbf{x}$ is located on the wrong side of hypersurface C_i. In case $\mathbf{x}$ is located on the right side of C_i, ϕ_i is updated such that $|\partial \phi_i / \partial t| > 0$. This is done so the color criterion can oppose the curvature criterion that will be discussed in the sequel, to prevent the hypersurface from showing wiggling behavior when these criteria contradict and alternate dominance in subsequent iterations.

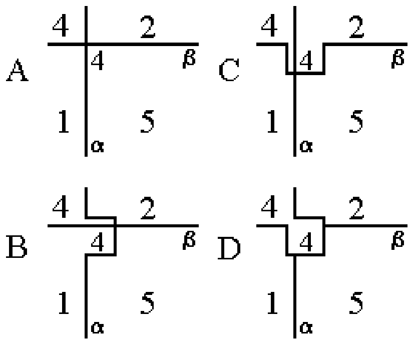

Figure 3. Near intersections of hypersurfaces, the CV algorithm might fail where the K-means algorithm does not.
The figure shows a hypothetical situation where four classes/subdomains are separated by two hypersurfaces α and β. The averages in the classes are 4,1,2 and 5. A pixel/object with value 4 is currently assigned to the class with average 5 (A) and should be assigned to the class with average 4 (D), which means that both hypersurface α (B) and hypersurface β (C) will have to move. For the obvious choice of parameters $\lambda_{i,j}$, the CV algorithm does not move the hypersurfaces, because $(4\text{-}5)^2<(4\text{-}1)^2$ and $(4\text{-}5)^2<(4\text{-}2)^2$. The K-means algorithm does move the hypersurfaces, because $(4\text{-}5)^2>(4\text{-}4)^2$.

Qualitative Evaluation Of The Algorithms

For undamaged, unblurred, synthetic images, all three algorithms (K-means, CV and CVK) work well. For natural

images or noisy images, the K-means algorithm cannot be used to completely segment the images [6,8], although it can still be useful to create an initial guess for other algorithms. CV and CVK are designed to handle these images as well [4,8,10].
Because

$$\mathbf{n} = \frac{\nabla \phi_i}{|\nabla \phi_i|} \tag{19}$$

is a unit normal to hypersurface C_i,

$$\kappa = \nabla \cdot \left(\frac{\nabla \phi_i}{|\nabla \phi_i|} \right) \tag{20}$$

can be used to calculate the curvature κ of the hypersurface. The hypersurfaces are moved by two effects; a fitting term makes sure that pixels in the same object have similar color and a curvature term makes the contours move in the direction that minimizes the curvature, $\frac{\partial}{\partial t}|\kappa| \leq 0$. Small objects have large curvature, while large objects have smaller curvature. Noise is actually made of many very small objects and have large curvature. The parameter μ can be set by the user to specify whether large or small objects should be detected and can be used to make the algorithms (CV and CVK) robust to noise. The curvature term deals with noise and keeps detected objects from being scattered.
The improvement in quality of the CVK over CV can not be measured in terms of correct output of the algorithms, but in terms of user friendliness; both algorithms produce correct results if the right set of parameters μ and $\lambda_{i,j}$ are put in. However, it can be tricky to tune the parameters μ and $\lambda_{i,j}$ for the CV algorithm, whereas CVK works fine by just choosing μ and setting all parameters $\lambda_{i,j}$ equal to one [8,10]. So the CVK algorithm is more robust in usage.

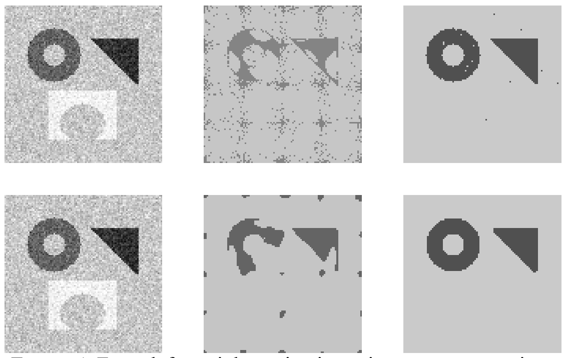
Figure 4: From left to right: noisy input image, segmentations after some iterations and final segmentation. Top: only K-means criterium was used. Bottom: both K-means and curvature criteria were used. CVK: $\lambda_0 = \lambda_1 = 1$, CV: $\lambda_0 < \lambda_1$ [6].

2. CVK ALGORITHM

Smaller Domain Versions

Full domain

Choosing level set functions to represent hypersurfaces introduces the flexibility that is so much needed, because hypersurfaces may split or merge while moving. On the other hand, they come with extra calculation time, because d-1 dimensional objects are represented by functions on a d dimensional domain.

Narrow band

The first step in reducing the time complexity is to acknowledge that much work is done in vain. In the digital model, a hypersurface moves over a pixel if the level set function on that pixel changes sign in the iteration. Assuming that far away from the hypersurface this changing sign does not happen, it is a waste of time to update the level set function there using the evolution equation. However, this assumption might not always be justified. Instead of updating the level set functions on every grid point, a speedup will be achieved if only values on grid points near the hypersurfaces are updated. By specifying a maximum distance δ to the contours and only updating the level set functions on grid points within this distance, a band-shaped domain is created.
Applying the narrow band method to the segmentation algorithm may cause the algorithm to fail. The narrow band method will produce correct results if the speed in normal direction depends only on local properties like the curvature. The fitting term may cause new hypersurfaces to appear out of nothing. This means that new hypersurfaces that should appear more than δ away from existing hypersurfaces do not get a chance in the narrow band method. Or objects that are not yet detected might not be detected at all if they are located too far from objects that are already detected.
The narrow band method can still be used with the segmentation algorithm if the initial hypersurfaces are chosen well. The algorithm can be expected to succeed if the union of the narrow bands corresponding to the initial contours cover most of the image domain Ω. In that case no speedup can be expected in the first few iterations after initialization.
Here the location of the narrow band is stored, along with the location of the zero level set, by the level set function; the edges of the band are the δ and -δ level sets. Reinitialization is needed to keep the distance between the level sets constant. In [1] a data structure is built that can store the location of the band during more than one iterations.

Multiresolution

Decreasing the resolution of an image decreases the size of the image domain and thereby reduces the time complexity of the segmentation algorithm. Changing the resolution of a digital image means that the same image is spread over a different number of pixels. A multiresolution method can take advantage of this. The multiresolution method should not be confused with the standard multigrid method, in which an iterative solution and the corresponding problem are coarsened to another grid, where the problem is solved and interpolated back to the fine grid. The grids are used recursively and iteratively. The multiresolution method for the image segmentation problem uses lower resolution versions of the original image to find initial solutions for higher resolution problems instead. So where the multigrid method starts at the highest level, returns to the highest level and uses all coarse grids regularly, the multiresolution method starts at the lowest level, ends at the highest level and uses all coarse grids only once.
The only required addition is a mechanism that can resize a d dimensional grid. If d=1, a value on a new grid point can be calculated by linear interpolating the values on the neighbor

grid points in the old grid (for the level set functions) or by copying the value on the nearest neighbor grid point of the old grid to the new grid point. If d>1, this mechanism is used for every dimension.

Quantitative Evaluation Of The CVK Versions

In the narrow band method, time is saved because calculations are only performed on a small domain. On the other hand, extra administration is needed to calculate and store the location of the narrow band. In the current implementation, the narrow band does not result in speedup but some speeddown, whereas previous versions of the narrow band did result in speedup. This is not a flaw in the current implementation of the narrow band method. In previous implementation of the full domain method, the level set methods had to be reinitialized after every iteration. This could be eliminated in the current implementation of the full domain method, but not in the implementation of the narrow band method.

The multiresolution method does not only reduce the number of grid points, but also reduces the number of operations that have to be performed on every grid point. For stability reasons, $\tau = O(h^2)$, with τ the time step and h the spatial step. On a coarser grid, larger time steps can be made, so lesser iterations are needed.

3. PARALLELIZATION

Parallel CVK Algorithm

In the sense of tasks that have to be performed, the segmentation algorithm is clearly sequential by nature. Therefore a data parallel model of computation is chosen. The first dimension of the domain is partitioned while the other dimensions are not partitioned. So if $\Omega = \{0,\cdots,\omega_0 - 1\}\times\cdots\times\{0,\cdots,\omega_{d-1}-1\}$, then $\{0,\cdots,\omega_0-1\}$ is partitioned into S subsets $\{0,\cdots,\omega_0-1\} = \{\varpi_0,\cdots,\varpi_1-1\}\cup\cdots\cup\{\varpi_{S-1},\cdots,\varpi_S-1\}$, where

$$\varpi_i = i\left\lfloor \frac{\omega_0}{S} \right\rfloor + \min\{i, \omega_0 \bmod S\} \qquad (21)$$

and

$$\overline{\Omega}_i = \{\varpi_i,\cdots,\varpi_{i+1}-1\}\times\{0,\cdots,\omega_1-1\}\times\cdots\times\{0,\cdots,\omega_{d-1}-1\} \quad (22)$$

If $d = 2$ this is called striped partitioning; the domain is a matrix and the columns are assigned to sub-matrices. In 3D it could be called sliced partitioning. To let terminology stay valid for $d > 2$ let

$$\{i\}\times\{0,\cdots,\omega_1-1\}\times\cdots\times\{0,\cdots,\omega_{d-1}-1\} \qquad (23)$$

be the (i+1)-th column of Ω. The first $\omega_0 \bmod S$ subdomains get $\lfloor \omega_0 / S \rfloor + 1$ columns and the other subdomains get $\lfloor \omega_0 / S \rfloor$ columns.

The grid points of the subdomains are assigned to S processes, a processor can run more than one process. Assigning a grid point to a process means that all data associated with that grid point is stored in the memory of the processor on which the process will run and that operations on the data will be performed by that processor. For every column that does not belong to the same process as its neighbor column an extra column is assigned; operations on this data are done by the neighbor process and data is updated during a synchronizing step between processes. In the synchronization step data is sent between processes, although sending is not the correct word if both processes are located on the same processor. This mechanism makes sure that subdomains overlap by one column, which is exactly enough for the scheme used in the discrete evolution equation.

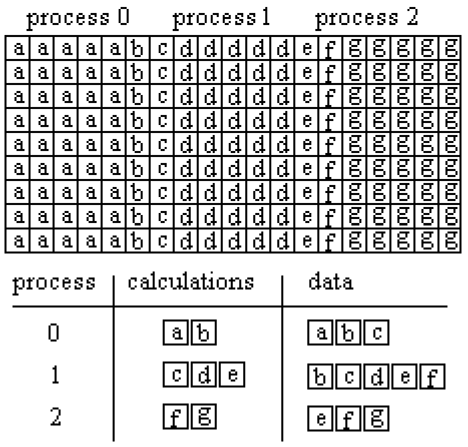

Figure 5. 2D domain distributed to three processes. Data associated with a subdomain is assigned to the concerning process and calculations needed to update the data are done by the process. Columns of data from neighbor processes are needed, so synchronization must take place.

Except after initialization, the hypersurfaces will be close to the objects to be detected. Since they are not distributed uniformly over the image, neither will be the narrow bands. Some parts of the domain may require more calculation than other parts. To distribute the work equally among the p processors, the S subdomains are mapped cyclically to the processors. Given p, S must be chosen so that $S \bmod p = 0$. Let b=S/p, then the domain is divided into b blocks and every block is distributed over p processors. So if p is given, b can be chosen small to reduce communication between processors or b can be chosen large to ensure a good load balance. In case no narrow band is used, the load is well balanced automatically, so b can be chosen equal to one.

Synchronous And Asynchronous Communication

Before every iteration the overlapping data must be synchronized. This requires S-1 exchanges of columns between neighbor processes. Also two all-to-all broadcasts must take place to evaluate the stop criterion and to recalculate the average intensities $c_{i,j}$.

Subdomains $\overline{\Omega}_i$ are assigned to processor $i \bmod p$. If neighbor processes are not located on the same processor, which is the case for every process if $p > 1$, synchronization must take place. This takes either two or three steps, depending on whether p is odd or even. In every step a process communicates with either its left or its right neighbor process. Here the communication time is minimized by bundling b overlapping areas into one message. Messages do not actually have to be copied to a combined message; instead computer memory can be organized so blocks of data that have to be sent in the same communication step are always located at consecutive addresses.

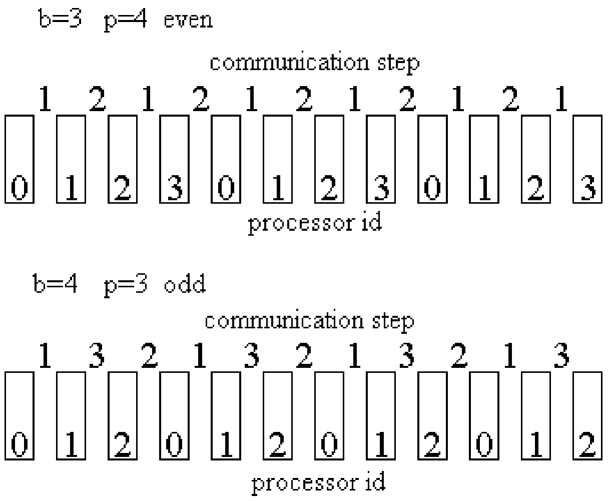

Figure 6. Diagram of data synchronization.
Top: 12 processes are mapped to 4 processors.
First step: communication between 0 and 1 and between 2 and 3.
Second step: communication between 0 and 3 and between 1 and 2.
Bottom: 12 processes are mapped to 3 processors.
First step: communication between 0 and 1.
Second step: communication between 0 and 2.
Third step: communication between 1 and 2.

Asynchronous communication

On parallel computer systems where communication is slow, run time could be optimized if calculations and communication could be done in parallel. The advantage of communicating asynchronously is that processors do not have to wait until the other processor involved in the communication is ready. The disadvantage is that after the communication is initiated, the data involved is accessible but not yet usable. This can be dealt with by choosing the order in which grid points are updated carefully: first non-overlapping grid points must be updated, then the overlapping grid points.

Iteration with synchronous communication:

- initiate synchronous communication of overlapping data
- wait until communication is finished
- update overlapping grid points
- update non-overlapping grid points

Iteration with asynchronous communication:

- initiate asynchronous communication of overlapping data
- while communication is in progress, update non-overlapping grid points in parallel
- wait until communication is finished
- update overlapping grid points

In MPI implementations, synchronous and asynchronous receive operations are implemented by the functions MPI_Recv and MPI_Irecv respectively. Synchronous and asynchronous send operations are implemented by the functions MPI_Send and MPI_Isend. MPI_Isend is in fact asynchronous, but MPI_Send can be synchronous or asynchronous, depending on the size of the systems message buffer; if the buffer is large enough, asynchronous sending is used.
Because some versions of MPI always use asynchronous sending, sometimes few improvement can be observed by using MPI_Isend and MPI_Irecv instead of MPI_Send and MPI_recv.

Results

The CVK algorithm was implemented (full domain, narrow band and multiresolution) in C++ using MPI functions for communication.
For undamaged, unblurred, synthetic images, K-means can be expected to be very fast. The algorithm can finish in just a few iterations, because in these kind of images there is only a small amount of different colors and pixels are assigned to classes based on color and not on location: if it is decided that a pixel with some color should be in some class, than all pixels with the same color are assigned to that class in the same iteration. CV and CVK on the other hand are slower, because only pixels near hypersurfaces are reassigned. For natural images or noisy images, the speed of the K-means algorithm is irrelevant, because the algorithm is not applicable.
Calculation time for CV is discussed in [10]. For the CVK algorithm, calculation time was measured on a Cray T3E parallel computer [13] in Delft and the DAS-2 clusters [14]. The efficiency of the algorithm run on DAS-2 on several grey-valued and color images, ranging in size from 100×100 to 600×480, using one or two level set functions, using different initializations and using different numbers of coarse grids, is shown is figure 7. Improvement in efficiency can be observed for most test cases on DAS-2, if asynchronous communication is used. Figure 8 shows an example of typical improvement by the asynchronous version. For large images, much less improvement can be seen, but for these images communication overhead plays a less important role relative to the increased number of calculations. Because both MPI_Send and MPI_Isend are implemented asynchronously with default setting of MPI_BUFFER_MAX on the Cray T3E, the use of MPI_Send and MPI_Isend do not influence the efficiency much.

4. CONCLUDING REMARKS

In this paper we discussed the parallel implementation of a new image segmentation method (CVK), a method that was created by integrating a classic clustering algorithm (K-means) into a recently developed active contour model (Chan - Vese). The narrow band method and the multiresolution methods were attempts to decrease the size of the image domain and thereby the calculation time. The multiresolution method proved very useful for regular images and indispensable for large images. Because of reinitialization after every time step, the narrow band method could not compete with the full domain version.

Parallelization is useful for both small and large images. Efficiency decreases when extra processors are added, but this decrease is smaller for large images than for small images. The MPI 1.1 does not support dynamic allocation of resources during the algorithm. The MPI 2.0 standard will support dynamic allocation of resources, which will make the multiresolution method more efficient; every time the algorithm moves from a coarse grid to a finer grid, more processors could be added.

Replacing synchronous communication functions with asynchronous ones made the DAS-2 [14] more efficient for most test cases. Efficiency did not change much for very large images. On the Cray T3E [13] that was used, no improvement in efficiency could be detected. Communication on the Cray is much faster than the DAS-2, but for the calculation time vice versa.

A possible future improvement could be adding more K-means optimizations to the algorithm. The average colors could be updated after every pixel is reassigned (on-the-fly K-means algorithm) or pixels can be randomly picked for reclassification instead of updating the full domain (R-means algorithm), however this makes parallelizing less efficient. Many initialization methods, like histogram based initialization, have been explored to improve the quality of the solution of the K-means algorithm or to decrease its calculation time and should be tested with the new segmentation algorithm.

5. ACKNOWLEDGEMENT

This report was presented and published in the Proceedings of the 2002 International Symposium on Distributed Computing and Applications to Business, Engineering and Science in Wuxi, China. Wuhan University of Technology Press, Wuhan, China, 7-5629-1881-3.

6. REFERENCES

[1] D. Adalsteinsson & J.A. Sethian, A fast level set method for propagating interfaces, Journal of Computational Physics 118, 269 (1995)

[2] K.R. Castleman, Digital image processing (Prentice Hall, New Jersey, 1996)

[3] T.F. Chan, B.Y. Sandberg & L.A. Vese, Active contours without edges for vector-valued images, UCLA CAM report 99-35 (1999)

[4] T.F. Chan & L.A. Vese, Active contours without edges, UCLA CAM report 98-53 (1998)

[5] T.F. Chan & L.A. Vese, Variational image restoration & segmentation models and approximations, UCLA CAM report 97-47 (1997)

[6] M. Leeser, K-means algorithms for unsupervised classification, http://www.ece.neu.edu/groups/rpl/projects/kmeans/ (1999)

[7] S. Osher & R.P. Fedkiw, Level set methods, UCLA CAM report 00-08 (2000)

[8] J.R. Rommelse, High performance algorithms in image segmentation, MSc thesis, Delft University of Technology (2002)

[9] J.A. Sethian, Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision and materials science (Cambridge University Press, Cambridge, 1999)

[10] L.A. Vese & T.F. Chan, Image segmentation using level sets and the piecewise constant Mumford and Shah model, UCLA CAM report 00-14 (2000)

[11] L.A. Vese & T.F. Chan, Reduced non-convex functional approximations for image restoration & segmentation, UCLA CAM report 97-56 (1997)

[12] W.L. Wan, Scalable and multilevel iterative methods, UCLA CAM report 98-29 (1998)

[13] High performance applied computing, http://www.hpcn.tudelft.nl/ (2000)

[14] The distributed ASCI supercomputer 2 (DAS-2), http://www.cs.vu.nl/das2/ (2002)

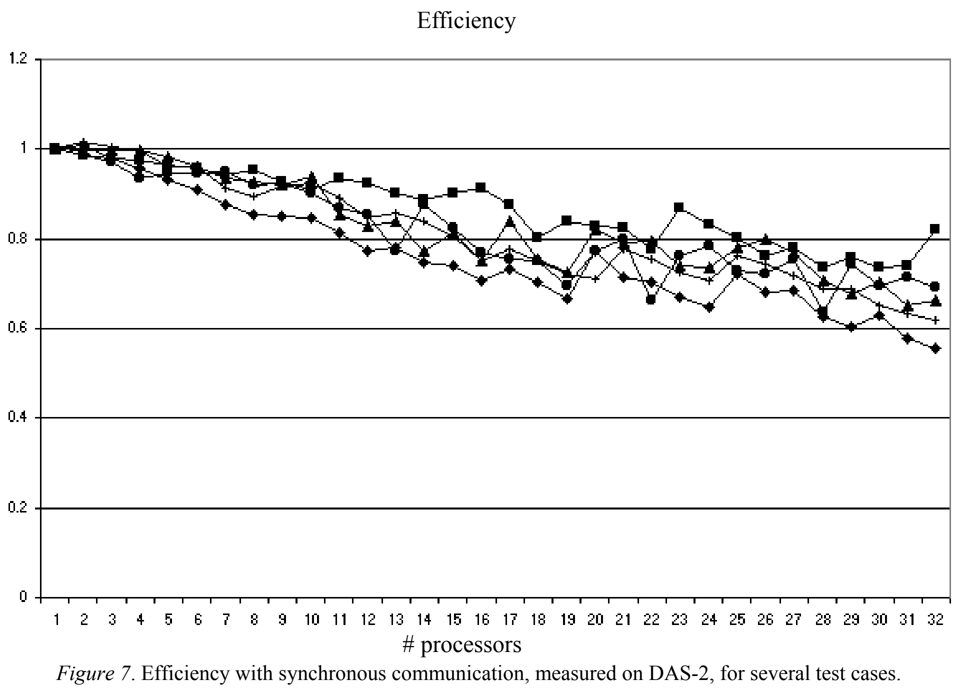

Figure 7. Efficiency with synchronous communication, measured on DAS-2, for several test cases.

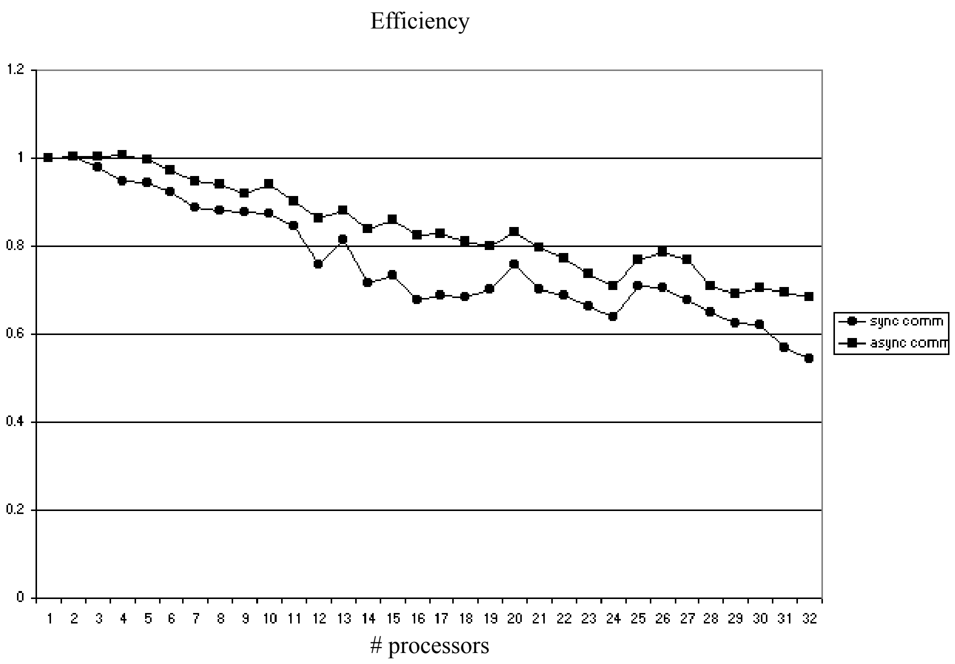

Figure 8. Efficiency improves if asynchronous communication is used on DAS-2, for some test cases.