Colour and Texture Based Pyramidal Image Segmentation

Milos Stojmenovic, Andres Solis-Montero, Amiya Nayak
[1] *University of Ottawa, Canada*
e-mails: mstoj075@site.uottawa.ca, amon@site.uottawa.ca, anayak@site.uottawa.ca

Abstract

The goal of image segmentation is to partition an image into regions that are internally homogeneous and heterogeneous with respect to other neighbouring regions. We build on the pyramid image segmentation work proposed by [BHR] and [SSN] by introducing a mixture of colour and texture cues in order to more accurately group regions. Statistical comparison of each colour channel separately along with edge intensity and orientation histogram comparison were the cues used for region merging. The input image size is no longer constrained as in [BHR], [SSN] and other similar regular pyramid based approaches due to a modification of the pyramid construction rule. The new algorithm is tested on a set of images from the Berkeley image segmentation benchmark set. Our algorithm is fast (produces segmentations within seconds), results in the correct segmentation of elongated and large regions, very simple compared to plethora of existing algorithms, and appears competitive in segmentation quality with the best publicly available implementations.

1. Introduction

Image segmentation can be very useful in many computer vision systems. It decomposes an image into homogeneous regions, which hopefully belong to the same object in the scene. Segmentation can be done according to some criteria by [MMB]. It is rarely achieved comprehensively for any single application, and algorithms that do perform well in one application are not suited for others.

Pyramid segmentation was proposed in [RH, TP], and further elaborated on in [BHR]. Pyramids are hierarchical structures where each level is built by computing a set of local operations on the level below (with the original image being at base level or level 0 in the hierarchy). Level L consists of a matrix of points where each point contains some data and a link to at most one parent point in level L+1. The value of a point at levels higher than the base level is derived from the values of all its children points at the level below. When this is applied to children points transitively down to the base level, the value at each point at a given level is decided by the set of its descendent pixels at the base level (its receptive field). Each point at level L also represents an image component and constructs the image segmentation at level L-1, consisting of pixels belonging to its receptive field (if nonempty). Thus each level has its predefined maximum number of components. However its minimum number of components is left open and depends on a concrete image.

Image segmentation pyramids can be classified into regular and irregular types. Regular pyramids have a well-defined neighbourhood intra-level structure, where only natural neighbours in the mesh that defines a level of a pyramid are considered. Inter-level edges are the only relationships that can be changed to adapt the pyramid to the image layout. A constant reduction factor between levels is found in literature and pertains to regular pyramids [MMB]. Regular pyramids can suffer several problems [A, MMB]: non-connectivity of the obtained receptive field, shift variance, or incapability to segment elongated objects.

To address the limitation of regular pyramids, irregular pyramids were proposed which vary in structure, increase the complexity and run time of the algorithms, and/or are dependent on prior knowledge of the image domain to be designed successfully. In the irregular pyramid framework, the relationships and reduction factors between levels are non-constant. Existing irregular pyramid based segmentation algorithms were surveyed in [MMB]. The described methods all appear very complex, have higher time complexities compared to regular pyramids and all consider the connectivity of the receptive field (base layer) as a design goal. Regular pyramids have a limited size of input: 2^n x 2^n pixels. We have modified the pyramid construction procedure such that it now accepts any size of input. Our new procedure eliminates this deficiency of regular pyramids while avoiding the complexity of irregular pyramid structures.

In [SSN], the authors observe that the connectivity of the receptive field (region in segmentation) is not always a desirable characteristic. For instance, in forestry and certain medical applications, the same type of vegetation or biomass could be present in several parts of the image, and treating them as a single segment may in fact be preferred for further processing. The proposed segmentation algorithm, called LS (Link Shifting), allows for non-connectivity of the receptive fields. Common receptive fields are coloured with common colours when the segmentation results are displayed.

The LS algorithm [SSN] is inspired by the regular pyramid linked approach (PLA) originally proposed by Burt et al. [BHR]. In the method proposed by [BHR], nodes from one level may alter their selection of parent at each iteration of the segmentation algorithm, which differs from previous pyramid structures. Starting from the base level (0), links between the current level *L* and level *L*+1 are decided. Each vertex at level *L* has four fixed candidate parents, and chooses the one which is the most similar from the higher level. The value of each parent is recalculated by averaging the values of its current children. Such iterations continue until the child-parent edges do not vary, or a certain number *T* of iterations is reached. The process then continues at the next higher level pair of levels. The main advantage of this method is that it does not need any threshold to compute the similarity between nodes at the same level. In the original method [BHR], each node must be linked to one parent node. Antonisse [A] introduced 'unforced linking' which allows the exclusion of some vertices from linking, and the presence of small components in the segmented image. Let *include(u)* be *m* times the standard deviation of a 3x3 neighbourhood around the node *u* (*m* is a parameter, whose value is 2 for 95% confidence and 3 for 99% confidence assuming a normal distribution of pixel intensities). This function is used to find the most similar parent *q* to node *u*. If *u* differs from q by at most *include(u), u* links to *q*. Otherwise, *u* fails to link to *q* and becomes the root of a new sub pyramid [A]. [SSN] changed this rule because 'unforced linking' of *u* and its parent was unnecessarily dependent on the neighbours of *u* in the existing rule from [A].

Antonisse [A] also proposed path fixing which defines some arbitrary path from the image(base) level to the top level to be fixed. This was indirectly applied here by the initial selection of random numbers for each pyramid vertex which were used for tie-breaking. Antonisse [A] also proposed randomized tie-breaking: when potential parent nodes have the same values, the choice of the parent is randomized. The tie-breaking rule has been completely redesigned in [SSN].

Burt et al [BHR] limit the choice of parent to 4 fixed nodes directly above the child. This approach has contributed to more successful segmentation, but due to the limited selection of parents, elongated and generally large segments that cover a significant portion of the image are not considered 'joined'.

We strive to design an algorithm that would properly handle elongated objects, while not enforcing connectivity of the receptive field and preserving shift invariance (the stability when minor shifts occur). This algorithm should also have favourable time complexity and execution time (within seconds, depending on the size of the images), and overall simplicity, so that it can be easily understood, implemented, and used in practice.

To achieve these goals, [SSN] made some simple changes in the way parent nodes are selected in the regular pyramid framework, which resulted in major improvements in their performance, including reduced shift variability and handling elongated objects. Instead of always comparing and selecting among the same four candidate parent nodes, each vertex at the current level selected the best node among its current parent, its current parent's neighbours, and the current parents of its neighbouring vertices at the same level. 4 connectivity neighbours were used in their implementation.

This paper makes numerous changes to the LS algorithm [SSN] and Co-parent LS algorithm [SSN2]. The new algorithm proposed here is called CTLS (Colour Texture Link Shifting). We describe bellow our new algorithm in full, and outline changes made from CPLS [SSN2]. The main improvement was changing the features used, from greyscale intensities to colour and texture cues, which resulted in more accurate merging of regions to produce results more appealing to humans. This is due to the greater quantity of information that is available when looking at an image in 3 channels instead of one, and taking into account the orientations and intensities of the edges present when making a segmentation decision. We have expanded upon the Student T-test between greyscale (one dimensional) based regions to an RGB (three dimensional) T-test based on differences between corresponding channels within regions, derived from means, deviations, and receptive field sizes. All three channels between two candidate regions must satisfy the test in order for them to be successfully merged. In addition to the three channel based comparison method, we add texture features to the merging process which was derived from [AGBB]. Each pixel is represented by a histogram of edge intensities divided into k bins, each of which is $\pi/2k$

degrees in width. Edge intensities of each pixel are calculated by 3x3 Laplacian edge masks and placed into their corresponding bin according to their orientation. Chi squared distributions of edge intensity histograms of regions are also taken into consideration when making a merging decision. A 95% confidence interval is enforced when making linking decisions based on edge intensity histograms. This texture based decision carries equal weight compared to the colour based decision in determining which child regions link to which parents starting from the base level of the pyramid.

Experiments were conducted on images from the Berkeley image segmentation data set. Evaluating segmentation quality in imagery is a subjective affair, and not easily done. "The ill-defined nature of the segmentation problem" [MMB] makes subjective judgment the generally adopted method (existing quantitative measurements are based on subjective formulas). In general, it is not clear what a 'good' segmentation is [MMB]. Even though the Berkeley data set provides human results that outline where people think the general segmentation boundaries should be, it does not label segments as being part of the same object or not. Our algorithm provides both outlines of the segmentation results and labels the segments. The quality of the obtained segmentations appears visually satisfying for at least one level in each image. Our experiments are compared with several measures in literature that have available implementations and the results show that our method is comparable to the best existing algorithms.

2. Pyramid Image Segmentation Algorithm

Here we describe the proposed pyramid segmentation algorithm. The input to the algorithm is a colour, 3 channel (RGB) image of dimensions m x n pixels, for $m, n \geq 1$. Let $C(c, u)$ represent the colour intensity of channel c, belonging to pixel u, where c={R, G, B} or {1, 2, 3}, with $C(c, u) \in [0, 255]$. The output is the original image overlaid with the resultant segmentations at each level. In the pseudo code and discussion below, L is the level of the pyramid, L=0, 1 …, M, $M = \lceil \log(\max(m,n)) \rceil$. The bottom level ($L$ = 0) is a matrix of m x n pixels, representing the original image. Level L is a matrix with $\lceil m/2^L \rceil$ rows and $\lceil n/2^L \rceil$ columns, L = 0, 1, 2 … M. The top level M has one element.

2.1 Creating the initial image pyramid at each level

We describe and use the overlapping image pyramid structure from [BHR], but applied to any image size. Initially, the children of node $[i, j, L]$ are: $[i', j', L-1]=[2i+e, 2j+f, L-1]$, for $e,f \in \{-1,0,1,2\}$. There are a maximum of 16 children. That is, each $\{2i-1, 2i, 2i+1, 2i+2\}$ can be paired with each $\{2j-1, 2j, 2j+1, 2j+2\}$ to produce the coordinates of the 16 children. For $i=j=0$ there are 9 children, and for $i=0$ and $j>0$ there are 12 children. There are also maximum index values ($\lceil m/2^{L-1} \rceil$, $\lceil n/2^{L-1} \rceil$) for any child at level L-1, which also restricts the number of children close to the maximum row and column values. For $L= M$ there are four children of single node $[0, 0, M]$ on the top: $[0, 0, M-1]$, $[0, 1, M-1]$, $[1, 0, M-1]$, $[1, 1, M-1]$ since the minimum index is 0 and the maximum is 1 at level M-1. Two neighbouring parents have overlapping initial children allocations. Conversely, each child $[i, j, L]$ for $L<M$ has 4 candidate parent nodes (if they exist) $[i'', j'', L+1]=[(i+e)/2, (j+f)/2, L+1]$, for $e,f \in \{-1,1\}$, where integer division is used (see Figure 1). Pixels at the edges of the image have fewer parents to choose from. The average intensity of all possible children is set as the per channel initial intensity value of the parent v, for nodes at levels L>0.

2.2 Texture comparison based on edge intensity distributions

Laplacian intensities were calculated using 3x3 Sobel masks for each pixel. The orientations of each pixel are calculated from its intensity in both directions. Let P be the greyscale version of the input image, determined by taking a weighted sampling of the red, green and blue colour spaces: $P=0.212671*R + 0.715160*G + 0.072169*B$.

Let $P(i, j)$ represent the value of the pixel at point (i,j). Output edge orientation images X and Y in the x and y directions respectively, are computed as follows:

$$\begin{aligned} X(i,j) &= -P(i-1,j-1)+P(i+1,j-1)-2P(i-1,j)+ \\ &\quad +2P(i+1,j)-P(i-1,j+1)+P(i+1,j+1) \\ Y(i,j) &= -P(i-1,j-1)-2P(i,j-1)-P(i+1,j-1)+ \\ &\quad +P(i-1,j+1)+2P(i,j+1)+P(i+1,j+1) \end{aligned}$$

The image $R(i, j)$ is called a Laplacian image, and is defined as $R(i, j)=sqrt(X(i, j)^2 + Y(i, j)^2)$. The result of this operation is another greyscale image with a black background and varying shades of white around the edges of the objects in the image.

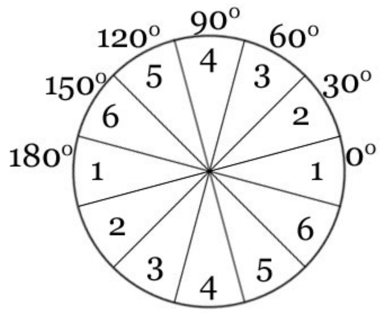


Figure 1 – Bin Orientations

The orientation of each pixel *R(i, j)* in the Laplacian image is found (in degrees) as

orientation(i,j)= $\arctan(Y(i,j), X(i,j)) \times 180/\pi$.

The orientations are divided into 6 bins so that similar orientations can be grouped together. The whole orientation space (2π) is divided into 6 bins. The division of the bins which places 0^0 or 90^0 at the border of two bins poses problems since all vertical and horizontal edges can fall into separate bins. We handle this problem by shifting all off by 15 degrees. This bin shifting technique was proposed in [S] in order to improve clustering of similar bin orientations. To fit all of the orientations into the 0-180^0 range, we add 180^0 if the angle is < 0^0, and subtract 180 if the angle is >180^0. The effects of these transformations can be seen in Figure 1.

Each pixel *u* in the original image, and each vertex *u* at each level of the pyramid, was assigned a vector of edge intensities *B*(*b*, *u*) (*b*= 1,2,…,6), where each field *b* in the vector corresponds to an edge orientation bin. At pixel level, *B*(*b*, *u*)= *R(i, j)* if *orientation(i,j)* is in bin *b,* and *=0* otherwise. As the image pyramid is constructed, the edge intensity vectors of parent nodes are maintained where each vector contains an average intensity *B*(*b*, *u*) per bin of all children belonging to it (calculation of averages is weighted by the size of receptive fields, similarly as for colors).

2.3 Testing similarity of two regions, unforced linking and the tie-breaking rule

Our algorithm makes use of a similar procedure to that of [SSN] for comparing the similarity of two regions, namely the receptive fields of a vertex *u* and its candidate parent *v*. In [SSN], these two vertices are similar (*similar(u,v)=true*) if their intensities are roughly the same. Also in LS [SSN], a simple threshold *S=15* for testing similarity was used. They also used a test for dissimilarity of two regions, to decide whether or not the best parent is an acceptable link, for the unforced linking option where a simple threshold based comparison against threshold value *D = 70* was used. Since we have a 3D space, we use the Euclidian distance between the points in RGB and define *S'* and *D'* as follows: *S'* = $\sqrt{3 \cdot S^2} = 51.96$ and *D'* = $\sqrt{3 \cdot D^2} = 121.24$.

In order to compute the similarity of edge distributions for vertices *u* and *v*, we used a normalized chi-squared distribution with a 95% confidence level and 6 degrees of freedom. To compute dissimilarity, the same distribution was compared to a confidence value of 5%, which corresponds to a value of ≈10. Let *n(u)* denote the number of pixels in the receptive fields of *u.* We also use same notation *C*(*c*, *u*) to denote the average color intensity of pixels in the receptive field of node *u*.

The two comparison functions are formally defined as follows.

Function *dissimilar(u,v)*

$$ColourDiff = \sqrt{\sum_{c=1}^{3} \left(C(c,u) - C(c,v)\right)^2}$$

$$BinsDiff = \sum_{b=1}^{6} \left(\frac{B(b,u) - B(b,v)}{B(b,u) + B(b,v)}\right)^2$$

If *ColourDiff > D'* ***Or*** *BinsDiff > 10* **then** *disimilar=true* **else** *disimilar=false.*

}

Function *similar* is defined via a statistical test between receptive field distributions *u* and *v* when possible. It resulted in better image segmentations, but when an analogous improvement was attempted for the function *dissimilar* there was no further improvement, so only the simple version was used. In case receptive fields exist with single pixels, and therefore variance values of 0, the simple colour intensity difference test was used with *S'* = 51.96. Let *n(u)* denote the number of pixels in the receptive fields of *u.* Note that *n*(*u*) is used in calculating *C*(*c*, *u*) from the colour intensity values of the children. For example, if w_1, w_2 and w_3 are children of *u* and *n*(*u*) = *n*(w_1) + *n*(w_2) + *n*(w_3), then

$$C(c,u) = \left(C(c,w_1)\cdot n(w_1) + C(c,w_2)\cdot n(w_2) + C(c,w_3)\cdot n(w_3)\right)/n(u),$$
$$c \in \{1,2,3\}$$

The intensity of the parent is the weighted sum of intensities of its children. Let *S(c*, *u*), where *c*={R,G,B} or {1,2,3}, denote the variances per channel of node *u*, that is, the variances of pixel color intensities in its receptive field.

Similar (u, v) {

If *n(u)* = 1 or *n(v)* = 1 {

$$colourDiff = \sqrt{\sum_{c=1}^{3} \left(C(c,u) - C(c,v)\right)^2}$$

```
        If colourDiff < S' Then similar=true Else
        similar=false}
    Else {
    Check = 0
    For each channel c {
        If n(u) <30 or n(v) <30
            $test = \frac{\left|C(c,u)-C(c,v)\right|}{\sqrt{\left(\frac{n(u)S(c,u)+n(v)S(c,v)}{n(u)+n(v)-2}\right)\left(\frac{1}{n(u)}+\frac{1}{n(v)}\right)}}$
Else
            $test = \frac{\left|C(c,u)-C(c,v)\right|}{\sqrt{\left(\frac{S(s,u)}{n(u)}+\frac{S(s,v)}{n(v)}\right)}}$
        If test > 2 then Check = check +1
    }
        $BinsDiff = \sum_{b=1}^{6}\left(\frac{B(b,u)-B(b,v)}{B(b,u)+B(b,v)}\right)^2$
    If check = 0 and BinsDiff < 2 then similar=true
    Else similar=false}
}
```

Each vertex *u* is initially assigned a random number *r(u)* in [0,1] which is never changed later on. Let *w* be a child node that compares parent candidates *u* and *v*, and let *better(w, u, v)* be one of *u* or *v* according to the comparison. The function is as follows.

```
        Function better(w, u, v)
        better=v;
        If similar(w,u) and similar(w,v) then
            { if n(u)>n(v) or (n(u)=n(v) and r(u)>r(v))
        then better=u }
        else if distance(w,u)<distance(w,v) or
            (distance(w,u)=distance(w,v) and
            n(u)>n(v)) or
            (distance(w,u)=distance(w,v) and
            n(u)=n(v) and r(u)>r(v))
        then better=u;
```

In our current implementation, the distance function is defined as follows (same as *ColourDiff* above):

$$distance = \sqrt{\sum_{c=1}^{3}\left(C(c,w)-C(c,u)\right)^2}\ .$$

The function *better* normally selects the parent that is closer to the child node, based on the *distance* function, which is currently their Euclidian distance in RGB space. However, if they are both close (that is, *similar*) to the child node then the decision is made based on the size of their receptive fields, which is used as the secondary key in the comparison. If needed, the random numbers are used as ternary, tie breaking key for final arbitrage.

2.4 Candidate Parents

Among the candidate parents, each vertex at the level below selects the one which the closest to it, using the function *better* described above. For the first iteration, each vertex has up to four candidate parents, as per initial setup described by [BHR], and seen in Figure 2 (a). This fixed set of candidate parents has been changed (for further iterations) in our algorithm by a dynamic flexible set of candidate nodes that revolves around the current parent selection and the parent selection of neighbouring vertices at the same level. Suppose node *w* = [*i*, *j*, *L*] is currently linked to parent *u* = [*i"*, *j"*, *L* + *1*] in iteration *t*, which we will denote simply by *p*(*w*) = *u*. The full notation would lead to *p*[*i*, *j*, *L*][*t*]=[*i"*, *j"*, *L* + *1*][*t*], and is convenient for easy listing of candidate parents. One set of candidate parents consists of the current parent and its 8 neighbours at the same level. Thus, in our notation, the candidate parents for the next iteration are: [*i"*+ *e*, *j"*+ *f*, *L* + *1*], where *e*, *f* ∈ {-1, 0, 1}. This produces a maximum of 9 candidate parents, which is a 3x3 grid centered at the currently linked parent. Four additional parent candidates were added by [SSN], by considering the current selection (from the previous iteration) of neighbouring vertices at the same level. This is illustrated in Figure 2 (b). For *w*=[*i*, *j*, *L*] and iteration *t+1*, we also consider *p[i+1, j, L][t]*, *p*[*i-1*, *j*, *L*][*t*], *p*[*i*, *j+1*, *L*][*t*], and *p*[*i*, *j-1*, *L*][*t*] as parent candidates, if they exist. Both sets allow us to shift the parent further away in the next iteration, and possibly link the current child to a remote parent after the iterative process stabilizes (with no more changes in the selected parents). This parent selection procedure is directly responsible for the ability of the algorithm to handle elongated objects. Note that we consider a 5x5 candidate parent grid centered on the current parent after half of the levels of the pyramid have been traversed by our algorithm. The segmentation quality was better than using 3x3 only or 5x5 only at all levels. This change does not however adversely impact the execution speed of the program since there are fewer children at higher levels of the pyramid, and raising the number of candidate parents from 13 to 29 does not constitute a significant increase in run time of the algorithm.

[SSN] introduced a concept called 'co-parent' identification in candidate parent selection. Its main purpose is to unite similar segments early on in the segmentation algorithm. Similarly, the co-parent of parent *u*, at level *L+1*, denoted *c(u),* is a node at the same level as *u*, is similar to *u*, and at least one child at level *L* switched from *u* to *c(u)* at the end of a parent selection iteration. Even if several co-parent candidates are available, at most one co-parent is selected by picking the one with the largest receptive field (the random number is used to break the tie if needed). Let *p(w)* be the parent for node *w* at the end of the previous iteration, and let *p'(w)* be the selected (possibly new) parent of *w* after comparing 13 or 29 candidate parents. The co-parent of *u*, denoted *c(u)*, is calculated using the function *find-co-parent(u),* who's pseudo code is below.

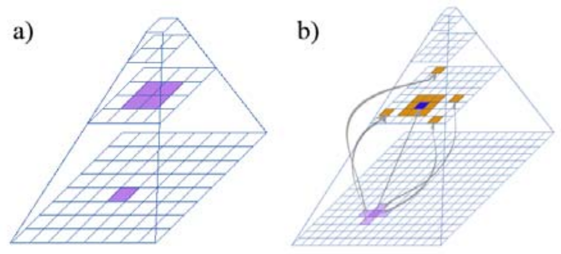


Figure 2 - Simple parent selection (a). 9 + 4 parent selection (b)

```
Function find-co-parent(u)  //  returns c(u)
c(u)=-1 // co-parent of u does not initially exist
For each child w at level L Do {  // p(w)=u
      If p'(w)  exists and p'(w) ≠ p(w)   {
          If similar(u, p'(w)) {
            If c(u)=-1 or (similar(c(u), p'(w)) and
               ((n(p'(w))>n(c(u)) or (n(p'(w))=n(c(u))
               and
               r(p'(w))>r(c(u))))
            then c(u)=p'(w) } }}
```

Once the co-parents of each parent are found at level *L*, each child tests its next iteration parent against the co-parent of its current parent. The better one of these two parents is selected as the next iteration parent.

2.5 Pyramid Segmentation

Once the pointers to parents have been initialized, the segmentation procedure may begin. Parent selections are attained at a given level (starting at level 0, and working toward the top of the pyramid) after a maximum of *T* iterations, before the process advances to the next level. At level *L*, each pixel initially points to the closest among four parents from the initial pyramid structure. In the subsequent iterations, it points to the (temporary) parent which best suits it in layer *L+1*, among the 9+4 or 25+4 candidate parents. This temporary parent is then compared to one more candidate, its co-parent, to yield the parent for the next iteration. The best parent is then tested for possible application of unforced linking based on dissimilarity. At the end of each iteration, the intensities of the parents in level *L+1* are recalculated based on the average intensity of the pixels in its current receptive field. Since these averages are calculated from the averages of its children, they must be appropriately weighted (by the number of pixels in the receptive fields of the children). Similarly the size of the receptive field, and the variance of the pixel intensities, are recalculated. Children that refused the link due to unforced linking (*unforced(w)=false*) are not considered in this calculation; however such children *w* continue looking for a parent in the next iteration. In case a child node has no current candidate parents, and its parent from the previous iteration has an empty receptive field, the child takes over that empty parent and transfers its receptive field onto it. This cycle of choosing parents, recalculating intensities, and reassigning parents continues for *T* = 10 iterations per pair of layers. The algorithm can be, at the top level, described as follows.

```
For levels L = 0 to N-1 Do {
    For each parent node u at level L+1 Do {
        Calculate initial parent intensity values,
        standard deviation, and receptive field size
        using 4x4 overlapping areas and default
        children. };
    For each child node w in level L Do {
            choose initial parent p(w)  among 4
            default parents in level L + 1.
        If dissimilar(w, p(w)) Then unforced(w) =
    false
                        Else unforced(w) =
                    true.  };
    For iter = 1 to T Do {
        For each parent node v at level L+1 Do {
            calculate new values for I(v), n(v), s(v)
        based on children u with
            p(u)=v and unforced(u)=true };
        For each child w at level L Do {
            select parent u among the 9+4 candidates
            (for L < (N-1)/2, and 25+4  otherwise)
            using method better (w, u, v) comparing
            currently best parent u and a candidate v.
            p'(w)= u   /* temporary parent  }
        For each parent u at level L+1 Do {
            find co-parents c(u) of u using function
            find-co-parent (u). }
        For each child w at level L Do {
            p(w)= better(w, c(p(w)), p'(w))  /*
            Compare p'(w) with co-parent c(p(w)) of
            its parent p(w) from the previous iteration
            to yield new parent p(w) for the next
            iteration;
            If dissimilar(w, p(w)) Then unforced(w)
        = false
                        Else unforced(w) =
                  true.  }
        }
}
Display segmentation for each level in pyramid.
```

3. Experimental Results

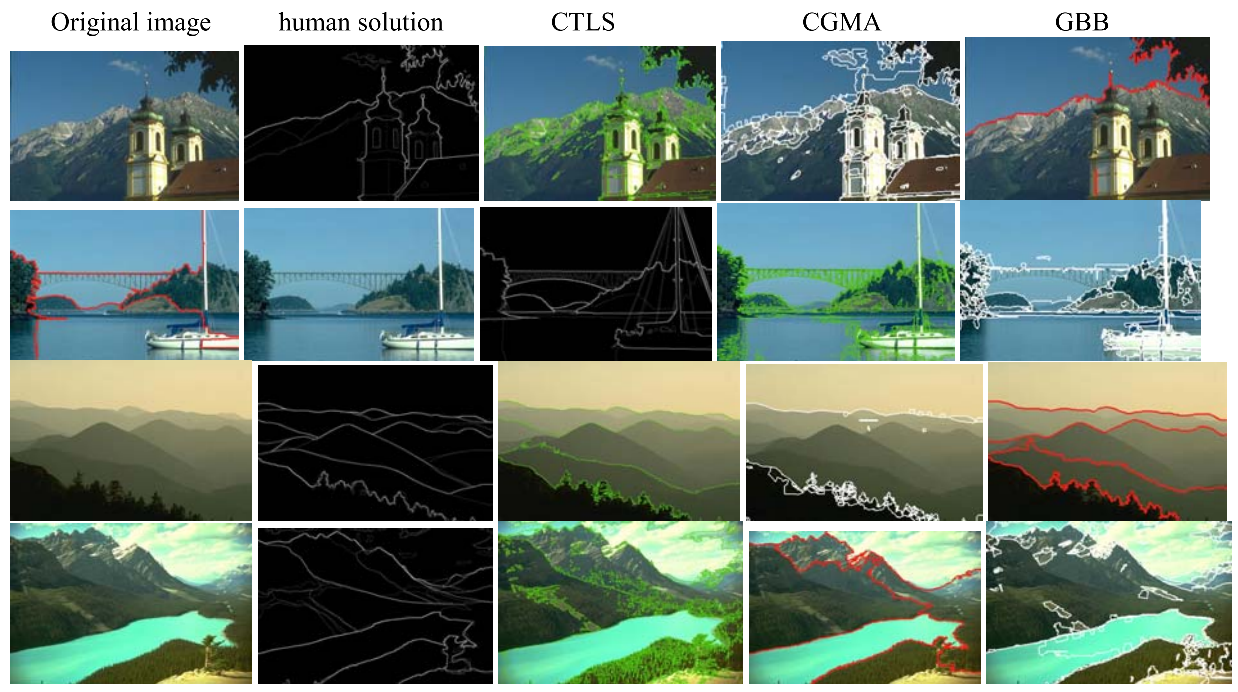
The algorithm presented here was designed to solve the problem of correctly segmenting objects in images within the framework of regular pyramid segmentation. It works on various types of everyday imagery: both colour and greyscale, although it was designed to take advantage of colour images which give more information with which to process the image. We have tested our algorithm on images of from the Berkeley image segmentation benchmark set, whose images are generally of size 481 x 321 pixels. The processing time per image is 35 seconds for the images on a single core of a Pentium 2.66 GHz dual core machine, implemented in C# on the Windows XP operating system.

3.1 Segmentation Results

We compared our Colour Texture link shifting (CTLS) algorithm to the algorithm proposed by [AGBB], who also employ a type of hierarchical segmentation structure but take into considerations texture as well as colour. The other algorithm used for comparison is the mean shift segmentation algorithm [CM] implemented by [CGM] and named EDISON. The human segmentation results of the images are also shown. All of the tested approaches (including our own) are relatively parameterless, or the parameters have been set once, and remain consistent throughout testing for all of the tested images. In the case of [AGBB, CGM], their default settings were used in the implementations found. We have also fixed parameter values in our own implementation, as described in the text. The test results of all the algorithms are seen in Figure 3.

We show only the best level of segmentation of each pyramid since they best reflect the desired segmentation results for these images. The algorithms shown here have tendencies either to oversegment or undersegment images systematically, but sometimes perform adequately according to human observations. The algorithm of [CGM] tends to oversegment images, and that of [AGBB] tends to undersegment them. Our CTLS algorithm tends to sometimes over segment areas that are textured. In has trouble with colour gradients since they contain no edges, but instead a gradual change in colour. Since our solution is threshold based, it eventually cuts this seamless gradient in peculiar locations. In the cases of coarsely textured images, [AGBB] performs best. However, based on this selection of images from [MFTM], the algorithms are fairly competitive.

.



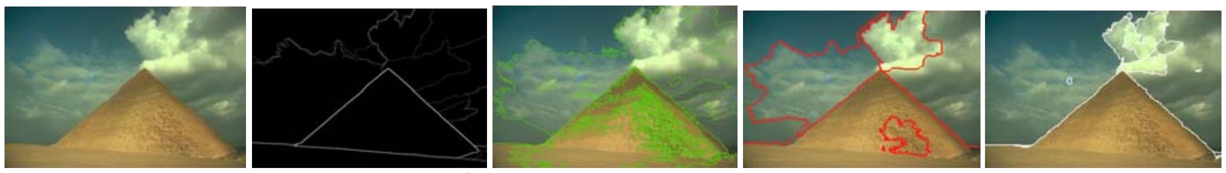


Figure 3 – Sample images and segmentations

4. Future Work

The part of the algorithm that needs most improvement is an automated selection of parameters that would increase the quality of segmentation of any general image. Currently some thresholds are set, but they are not optimal for all images. The introduction of a metric for self adjustment of parameters would be an interesting research topic. Another addition to the algorithm would involve finding better ways of joining child and parent nodes in the mid-level part of the pyramid. In the current scheme, pixels at the early levels of segmentation have many parents to choose from, and clusters are made fairly easily. However, these parents that represent early clusters find themselves relatively isolated with respect to the number of neighbours they have in their immediate vicinity. An adjustment needs to be introduced such that isolated parent nodes on any level can expand their search in order to be able to find neighbouring nodes with non empty receptive fields. This would enable their children to have a greater selection of parent nodes, and would speed up the segment merging process.

The algorithm can be modified in a variety of ways. We have tested a variety of options for functions *dissimilar* and *distance*, involving standard deviations, but none of them improved the outcome. However, there are other possible definitions for these functions that could be tested.

The algorithm can also be modified to enforce connectivity of receptive fields, either at the very end (applying a connected components algorithm to subdivide a region into connected pieces), or similarly splitting parents during the parent selection process.

To improve the outcome of this segmentation algorithm, one would have to have at least some prior knowledge of the scene that is to be segmented. Such knowledge includes the minimum possible segment size, and possibly a range of pixel intensities within a region that could be considered homogenous. Other solutions may include considering more than just greyscale intensities of input data. In the current implementation, just the RGB layers are considered, and they are combined into just a single layer greyscale representation of the original image. By considering the Euclidean distance between two 3D points in an RGB space instead of simply considering greyscale differences, more accurate parent selection could be achieved at the expense of increased computation time.

We have used and experimented with the overlapping image pyramid structure as originally proposed in [BHR]. This refers to the fact that parent vertices at level 1 have overlapping receptive fields. Antonisse [A] already argued that perhaps a non-overlapping structure could perform better. We left this modification for further study, so that we can first investigate the impact of a single major change proposed here, the use of flexible parent links.

References

[A] H. Antonisse, Image Segmentation in Pyramids, *Computer Graphics & Image Processing,* 19, 367-383, 1982.

[AGBB] S. Alpert, M. Galun, R. Basri, A. Brandt, Image segmentation by probabilistic bottom-up aggregation and cue integration, *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-07),* 2007.

[BHR] P. Burt, T. Hong, A. Rosenfeld, Segmentation and estimation of image region properties through cooperative hierarchical computation, *IEEE Trans. Systems, Man, and Cybernetics,* Vol. 11, No. 12, 1981.

[CGM] C. Christoudias, B. Georgescu, and P. Meer, Synergism in low level vision, *Proc. Int'l Conf. Pattern Recognition,* vol. 4, pp. 150-156, 2002.

[CM] D. Comaniciu and P. Meer, Mean shift: A robust approach toward feature space analysis, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, pp. 603-619, 2002.

[MFTM] D. Martin and C. Fowlkes and D. Tal and J. Malik, A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics, Proc. 8th Int'l Conf. Computer Vision (ICCV), Vol. 2, pp. 416-423, July 2001.

[MMB] R. Marfil, L. Molina-Tanco, A. Bandera, J. Rodriguez, F. Sandoval, Pyramid segmentation algorithms revisited, *Pattern Recognition,* Vol. 39, pp. 1430-1451, 2006.

[RH] E. Riseman, A. Hanson, Design of a semantically-directed vision processor, Tech. Rep. 74C-1, Department of Computer Information Science, University of Massachusetts, Amherst, MA, 1974.

[S] M. Stojmenovic, Real time machine learning based car detection in images with fast training, Machine Vision and

Applications (Springer), Volume 17, Number 3, pp. 163-172, August 2006.

[SSN] M. Stojmenovic, A. Solis-Montero, A. Nayak, Link shifting based pyramid segmentation for elongated regions, *Int. Symposium on Signal Processing, Image Processing and Pattern Recognition SIP 2009,* Jeju Island, Korea, December 10~12, 2009, CCIS 61 (D. Slezak et al. eds.), Springer-Verlag, 141-152, 2009.

[SSN2] M. Stojmenovic, A. Solis-Montero, A. Nayak, Co-parent selection for fast region merging in pyramidal image segmentation, *submitted for publication*, 2010.