Keywords: affine invariant regions, edge fragments, shape detector, interest points

Abstract: In this article we introduce a novel method for detecting multi-scale salient regions around edges using a graph based image compression algorithm. Images are recursively decomposed into triangles arranged into a binary tree using linear interpolation. The entropy of any local region of the image is inherent in the areas of the triangles and tree depth. We introduce twin leaves as nodes whose sibling share the same characteristics. Triangles corresponding to the twin leaves are filtered out from the binary tree. Graph connectivity is exploited to get clusters of triangles followed by ellipse fitting to estimate regions. Salient regions are thus formed as stable regions around edges. Tree hierarchy is then used to generate multi-scale regions. We evaluate our detector by performing image retrieval tests on our building database which shows that combined with Spin Images (Lazebnik et al., 2003), their performance is comparable to SIFT (Lowe, 2004). We also show that when they are used together with MSERs (Matas et al., 2002), the performance of MSERs is boosted.

1 INTRODUCTION

Computer vision algorithms for feature extraction have focused primarily on using Harris corners as interest points (Harris and Stephens, 1988) since they signify bi-directional change. This is followed by some criteria for corner selection (Bellavia et al., 2011) and then applying windows of specified scales to group information around them. Based on a similar approach, algorithms such as SIFT (Lowe, 2004) have solved much of the problem. Still this area is open to further research.

One of the challenges is the scale or the size of the window. Windows of predefined scale may not surround all the necessary the information. In order to address this issue, a scale pyramid is usually generated based on Lindberg’s diffusion scale space theory (Lindeberg, 1994) and some criteria for scale selection is incorporated.

A class of affine regions has evolved over time which tend to estimate a neighborhood covariant to affine transformations for either a corner, edge or any noticeable change in intensity. The approach to find affine regions vary from one another, such as maximally stable extremal regions (MSER) (Matas et al., 2002) seek the bounds of homogeneous stable regions following a watershed approach while edge based regions (EBR) (Tuytelaars and Van Gool, 2004) find seed points from Harris corners and follow the edges along the two directions of the corner to trace the boundary of the edges. All of the methods try to find either stable or pop out patches in a given image but due to different approaches, they end up in different parts of the same image producing regions varying in number, scale, repeatability and matching probability. In doing so, they tend to avoid scale space altogether by either selecting regions of an optimum scale (such as MSER, (Kadir et al., 2004)) or by generating multi-scale regions (such as Harris Affine and Hessian Affine regions (Mikolajczyk and Schmid, 2004)).

The performance of affine regions has been remarkably good with MSER performing the best (Mikolajczyk et al., 2005). Many recent algorithms perform slightly better, such as Wsh detector (Varytimidis et al., 2012), Medial Feature Detector (MFD) (Avrithis and Rapantzikos, 2011) and Boundary Preserving dense Local Regions (BPLR) (Kim and Grauman, 2011). Most of these algorithms tend to find either homogeneous regions, or regions enclosed by
common boundaries. They can replace each other, but since they seek similar image characteristics, though through different means, they can barely complement each other.

In the process of extracting features, the regions are mapped to a geometrical shape, usually an ellipse using the second moment matrix (Mikolajczyk et al., 2005). Unless they are of the same geometry, the shape of the boundary contributing to the affine bounds is lost in this process. If a fragment of the edge or boundary is enclosed in the ellipse representing the region, extracted descriptor may also include features for edge shape. But this is more a matter of chance than the design. Clearly, the regions representing edge fragments are more likely to complement such detectors and combined together, their performance can be further improved. This is shown by (Lazebnik et al., 2003) using Spin Image features for a combination of Harris-Affine regions (corner based) and Laplacian Blob detector for texture classification.

1.2 Our Approach

Our objective in this article is to estimate salient regions around edges to capture the edge shape. We do not use edge maps and instead we introduce a novel approach based on entropy. We use the fact that edges produce high gradients in their neighborhood and will therefore reflect such patterns in local entropy. Our approach is loosely based on Kadir-Brady Saliency detector (KBS). Salient regions as introduced by (Kadir et al., 2004) detect a region of distinguishable variation of entropy over scales. The performance of KBS has been low in terms of repeatability and efficiency (Mikolajczyk et al., 2005). Yet the approach is interesting.

As Kadir discussed in his thesis (Kadir, 2002) the close relationship between their approach and compression and segmentation algorithms in general. He suggested that the outliers of his method are low entropy and hence can be easily compressed. Following this lead, in this article, our work focuses on finding salient regions from the outliers of a compression algorithm instead which could potentially be inliers for saliency detection.

The compression algorithm chosen for this purpose is Binary Tree Triangular Coding (BTTC) (Distasi and Nappi, 1997). There are three main reasons for this choice; it is efficient, easily parallelizable and above all, it transforms the image into a tree structure which allows use of graph theory for image analysis. It has also been used for extracting multi-scale interest points for image retrieval task (Nguyen and Andersen, 2012). We will use it for salient region detection instead. Although, there are variants employing rectangular coding, we use triangular coding scheme since minimum non-zero area of a triangle is lowest than other geometric shapes and this enables localizing the edges more closely.

Our approach for salient region detection involves two steps; a top-down step of image decomposition into a rooted directed graph of triangles followed by a bottom-up step of synthesis combining filtered triangles into a region.

Distribution of this article is as follows. In section 2 we describe the method for image decomposition.
into a binary tree. A discussion about the tree depth and entropy variation in the graph is in section 3. Our method to filter image graph to find salient regions is described in section 4. Evaluation of proposed detector is done in section 5. Section 6 concludes the paper.

In this article, use of word face means triangle. One side of the triangle is an edge. A vertex is one of the three corners of a triangle. Branch always refers to the edge of a graph which is a link between two nodes. We find it necessary to define the terminologies since terms such as edges and vertices are used for both graphs and triangles, in general.

2 Decomposing Image Into Graph

In order to exploit graph theory for salient region detection, images must first be decomposed into a graph. We focus on triangular decomposition of image into a Binary Tree (B-tree) based on the approach followed by (Distasi and Nappi, 1997) for a lossy compression algorithm. In this method, low entropy or low complexity triangular regions are estimated by recursively triangulating an image. Planar linear interpolation is done between the vertices of a triangle using barycentric coordinates.

For a given image \( I = \{(x, y, c)|c \in \mathbb{C}, c = f(x, y)\} \) and \( C = \{c_a, c_b, c_c, \ldots\} \) is the set of intensity levels (0 to 255 for greyscale), the method proceeds by splitting the image into two isosceles triangles as shown in Fig. 1. Here \((x, y)\) specify pixel coordinates, and \(c\) is an intensity or color value. B-tree decomposition tries to approximate \( I \) with a discrete surface \( B = \{(x, y, d)|d = R(x, y)\} \), defined by a finite set of triangles. In this case, a triangle is a right-angled isosceles triangle. For any triangle with the vertices: \((x_1, y_1), (x_2, y_2), (x_3, y_3)\) and \(c_1 = f(x_1, y_1), c_2 = f(x_2, y_2), c_3 = f(x_3, y_3)\), we have a set \([x_i, y_i, c_i]|_{i=1,3} \in I\). The approximating function \( R(x, y) \) is computed by linear interpolation:

\[
R(x, y) = c_1 + \alpha(c_2 - c_1) + \beta(c_3 - c_1)
\]

\( \alpha \) and \( \beta \) are the barycentric coordinates, defined by:

\[
\alpha = \frac{(x - x_1)(y_3 - y_1) - (y - y_1)(x_3 - x_1)}{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)}
\]

\[
\beta = \frac{(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)}{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)}
\]

To check whether the estimated intensity value, \( R(x, y) \) is a suitable approximation to actual intensity value \( F(x, y) \), an error function is used:

\[
\text{err} = |F(x, y) - R(x, y)| \leq \epsilon, \epsilon > 0
\]

The idea is similar to mesh coloring scheme for triangular faces of a mesh using three colors values of the vertices. But meshes are created from point clouds and such a scheme tries to acquire more realistic appearance of the mesh faces. Image decomposition is somewhat the opposite, where a point set is being created from a real image and whether a given triangle fits well to the intensities in the triangle is decided by the smooth coloring criteria (eq. 2-4). Therefore, in this case, the intensity estimate \((R(x, y))\) is compared against the actual value \((F(x, y))\) for every pixel inside the triangle. If the error is higher than a global threshold for any pixel, the triangle is split into half along its hypotenuse. The image must be square sized (MxM) and this condition ensures all the triangles produced are isosceles right triangles. It runs recursively until error goes below the threshold or when the pixel level is reached, i.e. three vertices of a triangle are the three neighboring pixels.

The image decomposition process generates a binary tree, which is a rooted directed graph \( G(N, B) \) with \( N \) nodes and \( B \) branches. Following the properties of directed graphs, it has a root node which is the original square image itself. Each node has exactly two child nodes. Every node represents a triangle connected to its parent triangle through a branch, except the root node which has no parents. Parent nodes split into children nodes. Two triangles sharing the same parent are called siblings.

3 Tree Depth and Triangle Size

The recursive decomposition scheme of section 2 uses a global threshold of maximum allowable error for a given image, meanwhile trying to fit planar triangular surfaces. Therefore, more complex regions produce smaller triangles hence deeper nodes in the tree than relatively homogeneous regions. This results in a domain more triangulated near sharp changes in intensity. Consider the situation in Fig. 2 (b) where a black circle appears against white background and obviously, it has sharp intensity changes near the edge of the circle. The decomposition of this image, as expected, has produced bigger triangles in the homogeneous parts, while towards the edge of the circle, the triangles are getting smaller and smaller in size, until they reach the edge.

3.1 Twin Leaves

In this decomposition process, two important aspects are noticed: (1) Depth is shallow for leaf nodes not ending in high gradient regions and deep for leaf
nodes ending in high gradient regions. Node depth and triangle size are directly related, which means, shallow nodes represent bigger triangles as compared to deeper nodes.

By carefully examining the binary tree structure, we find two different types of leaf nodes, ones whose sibling decomposes into a sub-tree and others whose sibling does not further decompose. They are shown in Fig. 1, with cyan and red colors respectively. We call the red ones, twin leaves (as both siblings have similar characteristics), to distinguish them from the cyan leaf nodes.

3.2 Hierarchy of Entropy

In order to verify that the deepest nodes in the binary tree occur in the neighborhood of high gradients, we look at the local entropy characteristics. Starting from twin leaf nodes from different parts of Lena image (Fig. 2(a)), we walk up the tree towards the root node calculating Shannon entropy $H_C$, on the way for every triangle:

$$H_C(s) = -\sum_{c \in C} p_{c,s} \log_2 p_{c,s}$$  (5)

where $p_{c,s}$ is the probability density function (PDF) of the region of interest approximated by the histogram at any scale $s$. Obviously, as shown in Fig. 3(a), the entropy has to increase up the tree because the image decomposition process ensured that the tree must end in low complexity. The entropy constantly rises due to quickly involving greater part of the image. But a clue of information change or gradient would be change in entropy and not the absolute value of entropy. We therefore find difference between entropies using Kullback Leibler Divergence (KLD):

$$W_C(s_a,s_b) = \sum_{c \in C} \log \frac{p_{c,s_a}}{p_{c,s_b}}$$  (6)

$W_C(s_a,s_b)$ is a non-symmetric measure of difference between two PDFs or relative entropy for two consecutive non-overlapping scales $s_a$ and $s_b$. The product of KLD and entropy is:

$$Y_C(s_b) = W_C(s_a,s_b) \times H_C(s_b)$$  (7)

(Kadir et al., 2004) use the product of entropy peaks and the sum of absolute difference between entropies of the two scales producing the peak to estimate $Y_C$, their saliency measure. They consider scales with highest $Y_C$ to be salient. But in their case, scales increase gradually which results in a thorough but inefficient analysis. In our case, the scales double at every step giving steep rise to entropy due to which the product is also greater for increasing scales unless the other term, $W_C$ goes very low (Fig. 3(b),(c)). So $Y_C$ is not a reliable metric in our case. But a close look at the characteristics of $W_C$ curve reveals the very nature of the decomposition scheme i.e. highest entropy gain occurs for the twin leaves or their immediate parents and grandparents. This is an interesting result of the image decomposition. By picking up only the twin leaf nodes, we reach the high information regions of the image. They are in fact the points where the compression algorithm stops. Now, by using them a seed regions, we will estimate scale of information around them producing salient regions.

4 Salient Region Synthesis from Twin Leaf Nodes

Each node represents a triangle and two child triangles (nodes) combine to give a parent triangle (node). Therefore, whether we take the twin leaves or their immediate (first tier) parent node, its the same. We proceed with the first tier of their parents. Fig. 2 (b) shows a triangulated circle with first parents of the twin leaves displayed in red color. Again referring to the discussion in section 3.2, each one of these triangles provides an entropy peak in the local neighborhood and by looking at the image, it is quite obvious that their combination will enclose the complete
circle. By using the triangle vertices of the first tier of parents of the twin leaves as input, simple ellipse fitting will solve this problem and the scale of the fitted ellipse will provide an optimum scale for this image since it will enclose the entire circle. But this was a simple synthetic binary image. Images generally are much more complicated. Validity of the concept is proven on this image but we need to export the method to more complex images where the triangles form chains and groups. To estimate regions in such cases, we first need to group the triangles into clusters.

4.1 Graph Connectivity

As we have entered in the graph domain, so in order to find clusters, we explore the idea of graph connectivity from graph theory. One thing worth mentioning here is that graph connectivity is different from the general concept of connected components in image processing because in the later, two regions of an image are considered connected if they contain adjacent pixels. Since every node of our graph represents a triangle, so for us graph connectivity would means that the triangles are connected either through vertex or an edge which would mean that regions are connected only if they share one or more pixels.

Connectivity in graph theory is defined by the following properties for undirected graphs:

**Reflexive:** Any node $i$ is always connected to itself.

**Symmetric:** If a node $i$ is connected to $j$, then node $j$ is also connected to $i$.

**Transitive:** If node $i$ is connected to $j$ and $j$ to $k$, then nodes $i$ and $k$ are also connected to each other.

To find connected clusters of triangles, we first extract the selected twin leaf nodes out of the entire binary tree. Nodes are considered connected if the corresponding triangles share edges. Using these relationships, we zero initialize a $K \times K$ adjacency matrix, where $K$ is the number of selected nodes.

First we establish vertices connectivity relationship for the $K$ triangles. The adjacency matrix is then populated by adding 1 for every vertex connectivity between any two nodes (or triangles) to the corresponding matrix element. In this way, if two triangles share an edge, the corresponding value in the adjacency matrix is 2. This makes the adjacency matrix, sparse with very few non-zero elements.

There are many algorithms in graph theory to find connected components in graphs. We use Tarjan’s algorithm. Tarjan’s algorithm (Tarjan, 1971) tests the reachability of nodes starting from any random node using Depth First Search (DPS). The adjacency matrix is the input for this algorithm. It returns the connected components of the selected twin leaves. Ellipses are fitted to approximate the regions the groups represent using second moment matrix (Mikolajczyk et al., 2005).

4.2 Edge Connectivity Correction

Triangle representing twin leaves can be of different sizes, which means that they may share an edge, while sharing only one vertex. So the trivial notion of edge connection (two shared vertices) needs correction. In order to address this issue, we check all the pair of triangles sharing one vertex. We find the common vertex and then vectors of the edges of both the triangles from that vertex. Then we evaluate their dot and cross products. If the unit cross product is close to zero and dot product is greater than zero, we consider them edge sharing triangles of different sizes and hence mark it an edge connection.

4.3 Multi-scale Regions

Getting back to the discussion in section 3.2, we found out that $W_c$ peaks occur for twin leaf nodes and
their immediate parents. So far, we have only clustered 1st tier of parents of twin leaves which makes scale 1. In order to get multi-scale regions, we walk one level up the tree to the second tier parents of the same twin leaf nodes (or grandparents). Again, we follow the same approach and find the adjacency matrix and thereby the connected components. This gives us the scale 2.

We can do it as many times as does the tree depth allows us but we have noticed that after going up few levels, it becomes meaningless to continue further up as regions start merging distant edges. This is a clue that there must be a measure of scale selection which, at the moment, is not incorporated. In this article, for scales greater than 2, we use the connectivity at scale 2. This choice is based on observation.

Fig. 4 shows the Lena image fully decomposed and then with filtered twin leaves. Fig. 5 (c) shows Lena image with 1st (scale 1), 2nd (scale 2) and 3rd (scale 3) tiers of ancestors of twin leaves producing multi-scale regions. As one can observe, smaller ellipses come inside bigger ellipses (Lena shoulder and edge of hat) as does the children to parent and grandparents in the tree. Since the regions are based on twin leaves, we call them Twin Leaf Regions and use their acronym (TLR) in the figures. Detection of TLR’s for 3 scales takes only 70-100 ms on the Lena image (512x512) on an Intel Corei5 2.4 GHz, whereas KBS takes about a minute on the same system.

5 Evaluation: Image Retrieval Test

We evaluated the performance of TLR on our building database (Nguyen and Andersen, 2012), which consists of images of 21 buildings, each having 4 to 6 images with changing view point and scale (Fig. 7). For comparison, we used MSER regions which are among highest performing affine regions (Mikolajczyk et al., 2005) and SIFT based on Difference of Gaussian interest points (DoG) and scale space (Lowe, 2004). For MSER region detection, we use MATLAB’s Computer Vision System Toolbox (R2013a) implementation and for SIFT we use VLFeat’s implementation ( Vedaldi and Fulkerson, 2008), both with their default settings. We used the implementation of Mikolajczyk for extracting SIFT, Shape Context (SC) (Belongie et al., 2002) and Spin Image (SI) ( Lazebnik et al., 2003) descriptors from the TLR and MSER regions.

Since every image of the building is representative of its class, we followed the approach adopted by ( Lazebnik et al., 2003) for finding retrieval performance. We selected one image from the database and matched it with all the others. Best matches were decided based on correspondence voting. To establish correspondences, MATLAB’s Computer Vision System Toolbox (R2013a) implementation of Nearest Neighbor Ratio (Mikolajczyk and Schmid, 2005) was used with default settings. Ground truth was manually

\[ \text{Average Precision (AP}_{\text{all}} \]
assigned. Precision was calculated by the following relation:

\[
\text{precision} = \frac{\text{Number of correct matches so far}}{\text{Number of retrieved images}} \quad (8)
\]

Following this relation, precision was calculated for every correct match. Incorrect matches have zero precision. In order to find Average Precision (AP), we found the mean precision for each rank (retrieval number) in two different ways. One was the standard approach which includes only correct matches (AP\text{correct}). The other includes both correct (precision > 0) and incorrect (precision = 0) matches and therefore it penalized more the algorithm having a higher number of incorrect matches between two correct matches.

Table 1 shows mean average precision (MAP) for both the cases, only for the first four retrievals since this is the minimum number of images any category has in our dataset. It can be observed that the trends are almost similar, therefore, we will comment only on AP\text{all} and MAP\text{all} since this provides a more rigorous evaluation.

The AP\text{all} graph is displayed in Fig. 6. For comparison, the baseline is DoG+SIFT and its average precision is above 95% for the first correct retrieval. MSER+SI and MSER+SC also performs similarly. TLR has fairly low discrimination with SC. This is quiet expected since all the buildings have similar features near edges. But the performance of TLR with SI is very high, close to DoG+SIFT and MSER+(SI,SC).

We can observe that DoG+SIFT performs better than MSER and TLR whether with SI or SC. But when SI features of TLR are combined with SI features of MSER, then the simple 50 dimensional SI descriptor outperforms 128 dimensional SIFT vector. This is also shown in Fig. 6 with the red line. Although TLR+SC combined with MSER+SC give poor results, but MSER+SI combined with TLR+SC gives results better than either, individually. This validates our idea.

<table>
<thead>
<tr>
<th>Detectors/Descriptors</th>
<th>MAP\text{correct}</th>
<th>MAP\text{all}</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoG+SIFT</td>
<td>0.983</td>
<td>0.89</td>
</tr>
<tr>
<td>TLR+SI</td>
<td>0.982</td>
<td>0.85</td>
</tr>
<tr>
<td>MSER+SI</td>
<td>0.972</td>
<td>0.84</td>
</tr>
<tr>
<td>MSER+SC</td>
<td>0.983</td>
<td>0.85</td>
</tr>
<tr>
<td>TLR+SC</td>
<td>0.830</td>
<td>0.44</td>
</tr>
<tr>
<td>MSER+SC &amp; TLR+SC</td>
<td>0.932</td>
<td>0.69</td>
</tr>
<tr>
<td>MSER+SI &amp; TLR+SC</td>
<td>0.966</td>
<td>0.86</td>
</tr>
<tr>
<td>MSER+SI &amp; TLR+SI</td>
<td>0.987</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Table 1: Mean Average Precision (first 4 retrievals). MAP\text{correct} is the MAP of correct matches (precision > 0) only. MAP\text{all} is the MAP including both correct (precision > 0) and incorrect matches (precision = 0).

purpose of this article was to introduce a new approach for edge shape exploitation and highlight the shortcoming of affine regions detectors for homogeneous regions. In this article we address their outliers and show that this boosts their performance.

This research has indicated some new directions for further work, primarily making use of inherent flexibility of BTTC for parallelization which will be pursued in future.

6 Conclusion and Future work

In this paper we have presented our approach and the results of our Twin Leaf Region detector which detects salient around boundaries/edges. It performs well for image retrieval using simple low dimensional SI features. In addition to that, when combined with MSERs and SI, the performance is better than DoG+SIFT. It is highly efficient.

Our domain of application is agriculture and in that, edge shape is an important feature. The main

REFERENCES


Figure 7: TLR and MSER regions detected for two views of three different buildings. It can be observed that regions detected by both methods complement each other.


