An Accelerated Hierarchical Approach for Object Shape Extraction and Recognition

M.K. Quweider and Bassam Arshad

CS Department, University of Texas, RGV

Brownsville Campus, Texas 78520, USA

Mahmoud.Quweider@utrgv.edu

***Abstract*— We present a novel automatic supervised object recognition algorithm based on a scale and rotation invariant Fourier descriptors algorithm. The algorithm is hierarchical in nature to capture the inherent intra-contour spatial relationships between the parent and child contours of an object. A set of distance metrics are introduced to go along with the hierarchical model. To test the algorithm, a diverse database of shapes is created and used to train standard classification algorithms, for shape-labeling. The implemented algorithm takes advantage of the multi-threaded architecture and GPU efficient image-processing functions present in OpenCV wherever possible, speeding up the running time and making it efficient for use in real-time applications. The technique is successfully tested on common traffic and road signs of real-world images, with excellent overall performance that is robust to moderate noise levels.**

***Keywords*—** Automatic Object Recognition, Fourier Descriptors, Feature Invariance, GPU , OpenCV.

I. INTRODUCTION

Automatic object recognition is fundamental to many applications in image segmentation and retrieval, pattern recognition, remote sensing, computer vision, machine learning, and data mining, among many others. The recognition relies on extracting features, from the object of interest, that are usually invariant to translation, scale, and rotation. Features can be region-based (focused on the interior of the object), shape/contour based (focused on the boundary of the object), texture-based (focused on statistical properties of the object), or color/intensity based (focused on the color or intensity levels of the object). Additionally, the features can be viewed either in the spatial (captured mode) or in the frequency (spectrum) domain. The two domains are usually dual of each other, and provide complementary information.

In this paper, we extend the functionality of the frequency-domain based Fourier descriptors method, which is well known for handling images in the spectral domain and is ideal for object recognition, by creating a hierarchical representation of an object that takes into account any enclosed contours within the object's outer boundary. The paper has two contributions: first, we present a new representation model for objects of interest that is nested and hierarchical in nature along with appropriate metrics to capture the spatial relationship of the outer contour with the inner ones; second, we integrate the representation model within an automatic detection pipeline algorithm that is real-time, efficient, and takes advantage of any accelerating GPU functionality present at the hardware level. In section 2, we review object recognition and image features including Fourier descriptors. In section 3 we present the Fourier descriptor hierarchical model; section 4 presents the automatic detection algorithm in details. Section 5 presents the simulations results. Conclusions and future work are given in section 6.

II. AUTOMATIC OBJECT RECOGNITION

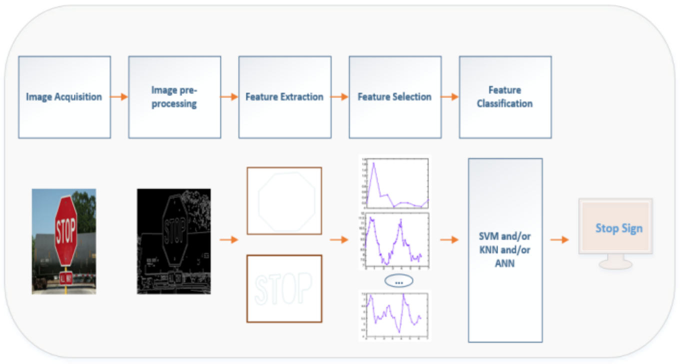


Fig. 1 Automatic Object Recognition Pipeline

As shown in Figure 1, automatic object recognition usually starts by acquiring and image and segmenting it into meaningful objects, and then extract relevant features for each object to form a feature vector that is later used for classification. The object recognition problem can be effectively divided into three separate sub problems: Feature Detection, Feature Description and Feature Classification. Feature Detection – The goal of this problem is to detect low-level image features. Features here are points or regions of interest, which for a known object will occur consistently across different images and scene, irrespective of changes in scale, translation and rotation.

Feature Description – The goal here is to find a robust mathematical description of the previously obtained feature, which will offset the effects of spatial and affine transformations, for similar feature. Also, a low-dimensionality feature descriptor is desired that is efficient to compute, and match, over a database of similar descriptors. Feature Classification – This step deals with the labelling or matching of query feature descriptors previously evaluated, across a database of already labelled feature descriptors. This step encompasses standard algorithms and techniques from the area of machine learning, as well as, basic matching methods specific to the feature descriptor being used.

Some of the most popular features used today include color, shape, corners/interest-points and texture, as reviewed in [1-3],[6-7],[10]. While color and texture features, offer good local and global discriminative properties, in order to describe the object accurately and invariantly with respect to scale, shift and rotation, spatial features like corners and edges have been vastly used [2]. Interest points based features, commonly built upon corners and blobs, detected across different scale spaces, and refined using non-maximum suppression techniques, account for a formidable portion of object detection methods being employed today. The Scale-Invariant Feature Transform (SIFT) by Lowe [3] is probably the best known and widely used technique, in this category. See [1-10] for a good references on local invariant feature detectors.

One of the most enduring algorithms for extracting features is the Fourier descriptors which is able to produce features that are invariant to translation, rotation and scaling by doing simple operations on them [7,10].These features (descriptors) are excellent to identify objects uniquely.

1) Fourier Descriptors

Fourier descriptors are based on extracting a set of two-dimensional points $\{(x(k), y(k)): k= 0,\ldots,n\}$ from the boundary contour points of the object. A shape signature (a 1-D representation of the 2-D region) is then created by transforming the representation into the spectral domain to create the descriptors, possibly with invariant properties. As shown in Figure 2, the process involves preprocessing the image which could include binarization, denoising, and boundary tracing and connection.

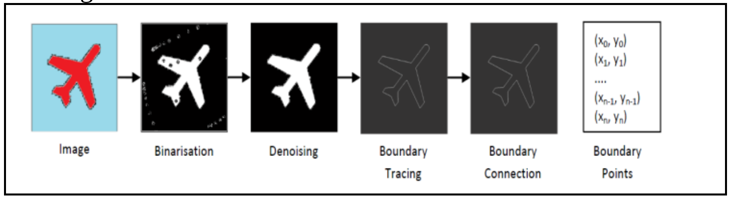


Fig. 2 Automatic Object Recognition Pre-Processing

The 1-D signature, *s(k),* is created from boundary pixels as a centroid normalized distance in the complex plane given by [10-12]:

$$s(k) = \sqrt{[x(k) - x_c]^2 + j[y(k) - y_c]^2} \text{ for } k = 0,1,\ldots,n\text{-}1$$

Where xc and yc represent the x-centroid and y-centroid, respectively, and n is the number of sampled boundary points. The Fourier coefficients are simply the Discrete Fourier Transform of the curve s(k).

$$a(u) = \frac{1}{K}\sum_{k=0}^{n-1} s(k) e^{-j2\pi uk/N} \quad \text{for } u = 0,1,\ldots,n\text{-}1$$

Fourier Descriptors, FDi, are usually constructed from a subset of the Fourier coefficients by ignoring higher frequency terms and normalizing the resulting coefficients to achieve invariance, knowing that general shape features are captured by the first few low frequency terms, while higher frequency terms capture finer details of the shape. Fourier descriptors, acting as extracted features from the detected object, can then be tested by means of similarity measures against other images' descriptors in order to identify it.

2) Hierarchical Fourier Descriptors

Although the Fourier descriptors method is an excellent choice to identify objects, it is limited to recognizing only the external contour of an object, and fails to acknowledge details within the object, that is its interior. The interior content, such as other contours, may help identify the object even further and more precisely. Figure 3 depicts the flaw of obtaining Fourier descriptors derived only from the object's outer contour.

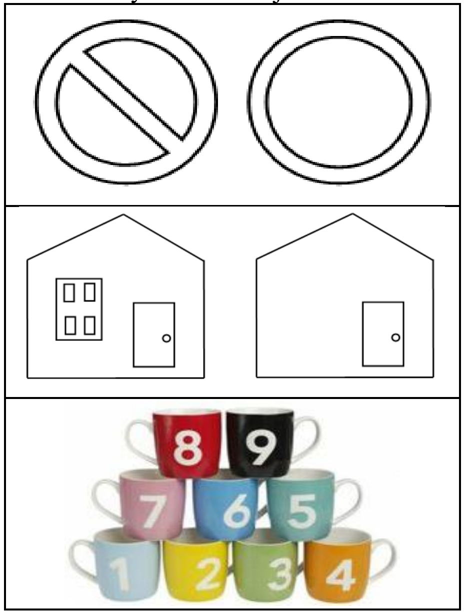


Fig. 3 Objects with identical Fourier descriptors

It would be more discriminant if the interior contours could be incorporated as part of the overall signature of the object which would sharpen the object's signature while increasing the precision of the recognition process. Therefore, our method proposes to take the usual Fourier descriptor signature process a step further by identifying the object, not only via a single external contour, but, in addition, acknowledging multiple internal contour Fourier descriptor signatures and making them part of the object's identity. The new methods is dubbed Hierarchical Fourier Descriptors, HFD. The HFD model is presented next.

III. HFD Model Representation

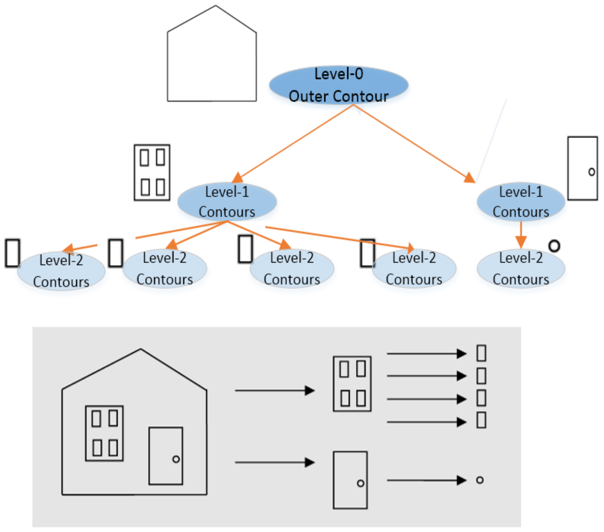


Fig. 4 Hierarchical Fourier Descriptors Model

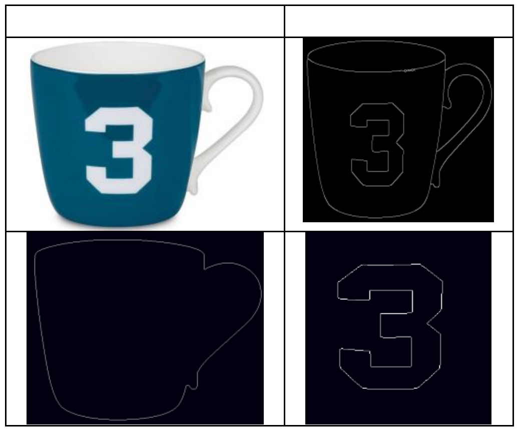


Fig. 5 Hierarchical Fourier Descriptors Model for a Cup

The proposed HFD model is designed as a tree-based data-structure in which the outer contour is set as the root of the HFD. Subcontours that meet certain criteria are considered as children of the parent contour. Theoretically, we can build the model to any desired depth; However, in reality, many objects are easily identified with one or two levels of subcontours. A detailed synthetic example that has three-level depth (including the root) is given in Figure 4. Figure 5 gives the contours, outer and inner, of a coffee mug. Interestingly, there can be many distinct cases that arise when having an object with multiple internal contours: (1) an object with multiple contours that may have internal nested contours, (2) an object enclosing multiple contours that are not nested (3) and an object with multiple single nested contours. Each case poses a distinct method of data structure and inherently distinct complexity times that determine the practicality of the method. Domain-specific problems should decide which case to emphasize.

We take a simple but effective approach in which the HFD is developed to extract candidate parent and child contours, based on their hierarchical relationships. The inherent spatial relationships between the outer and inner contours of an object can be accurately defined and matched against a stored template. We will inscribe these details into our feature descriptor, along with the FDs, to invariantly describe the closed contours of an object. As an offline one-time task, we will extract FDs from a database of training shape-images, that we have created separately, and store them in a serializable data structure. Classification of the queried FDs is performed against the offline database, and the nearest match is returned. The predicted classification shape-labels are then matched against stored models of pre-defined objects, in a hierarchical fashion. The final results are then presented.

A. *Recognition and Classification*

With the new HFD acting as a feature vector, we can feed those features into a SVM, a kNN an ANN, or any valid classification algorithm [11-15]. SVM has been used successfully to classify shapes, using FDs over the standardized MPEG-7 test database of various object shapes. Similarly, the kNN method has also been used quite frequently in order to classify FDs for the character-recognition.

Our HFD, consisting of the a concatenated set of samples descriptors for the parent and children contours was complemented with a set of spatial descriptors for the child contours with respect to the parent contours. In particular, we defined three metrics: child-parent contour centroid distance (normalized by parent perimeter), child contour aspect ratio and child contour orientation. The goal of these metrices is to add discrimination and invariance to the feature vectors when comparing objects at different scales, orientations, and rotations. Details of the classification are given in the simulation section.

```
Algorithm: Extract Parent and Child Contours
  Data: QueryImage
  Result: tuple <parentContour, childContours, childSpatialDescriptor> parentChildContours
  begin
    contours ⟵ findContours(QueryImage)
    for i ∈ contours do
      if candidateParentContour(contours(x)) then
        parentContour ⟵ contours(x)
      end
    end
    allChildContours ⟵ getAllChildContours(parentContour)
    for j ∈ allChildContours do
      if ( concentricToParent(allChildContours(j)) OR allChildContours(j) <areaThreshold ) then
        discard allChildContours(j)
      else
        childContours ⟵ allChildContours(j)
        childSpatialDescriptor ⟵ getSpatialDescriptor(allChildContours(j))
      end
    end
    parentChildContours ⟵ tuple <parentContour, childContours, childSpatialDescriptor>
  end
```

Fig. 6 Hierarchical Fourier Descriptors Model for a Cup

B. GPU Acceleration with OpenCV

As the algorithms for feature detection and extraction and object recognitions are complex and processing intensive, they can benefit immensely from the tools supplied by Nvidia's CUDA (Computer Unified Device Architecture) that provide the parallel computing at relatively low costs with a performance potential similar to supercomputing clusters [16]. As the benchmark simulations shown in Figure xx illustrate, using GPU-based algorithms can provide speeding factors in excess of 6 to 30 times over those of CPU-based counterparts. We have taken advantage of the highly scalable GPU architecture that comes with OpenCV to replace the traditional CPU-based algorithms with their GPU-based equivalent.

Many of the OpenCV GPU algorithms are written using CUDA, and since they are designed as host API extensions, they can be used if a dedicate Memory card is present, but can also be replaced by their CPU equivalent when it is not.

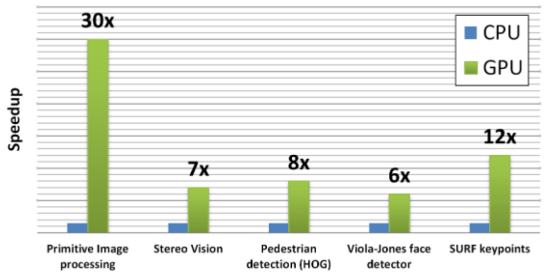


Fig. 7 Hierarchical Fourier Descriptors Model for a Cup

Most of the routines we used, take advantage of cv::gpu::GpuMat, which is a primary container for data kept in GPU memory. Because the GpuMat interface is very similar to the cv::Mat (its CPU counterpart), the acceleration is almost seamless. The heart of acceleration is a result of the ability to invoke several GPU algorithms without downloading data, because many GPU functions receive GpuMat as input and output arguments. The accelerated pipeline used the following GPU-accelerated computer vision functions in the implementation: gpu::dft; gpu::convolve; gpu::threshold; gpu::warpAffine; gpu::warpPerspective; gpu::bilateralFilter; gpu::Canny; and gpu::knnMatch¶

IV. ALGORITHM

Having identified all the individual components in the previous sections, we present the general steps for the algorithm. The simulations section gives the details of the algorithm with results on real-life images.

1. Create a new feature extraction model/method, to detect closed object parent-child contours, of interest/significance.
2. Derive the normalized FDs for the obtained object contours.
3. Create a database of FDs from a collection of known shapes at different scales, translation and rotations, to facilitate the effective classification of the FDs, in the later steps.
4. Match the derived FDs against the database of labeled FDs, and obtain the closest classification.
5. Develop an algorithm to match the individually classified contours against a stored template of the object model.
6. Implement multi-threading and GPU-based functions where possible, to accelerate the entire process.

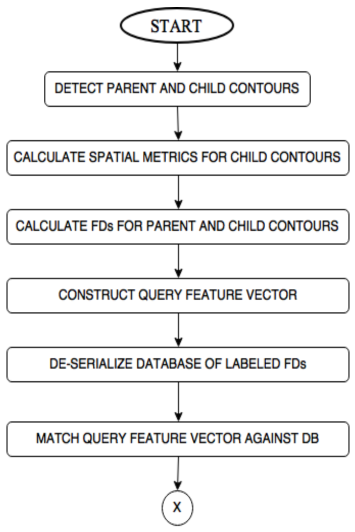

START
DETECT PARENT AND CHILD CONTOURS
CALCULATE SPATIAL METRICS FOR CHILD CONTOURS
CALCULATE FDs FOR PARENT AND CHILD CONTOURS
CONSTRUCT QUERY FEATURE VECTOR
DE-SERIALIZE DATABASE OF LABELED FDs
MATCH QUERY FEATURE VECTOR AGAINST DB
X

Fig. 8 Block Diagram of the Proposed Hierarchical Fourier Descriptors Algorithm

V. SIMULATIONS

We selected some of the most common traffic and road signs, as a test case, for our object recognition pipeline. As an offline one-time task, the algorithm extracts FDs from a database of training shape-images, that we have created separately, and store them in a serializable data structure. Classification of the queried FDs is performed against an augmented offline database, and the nearest match is returned. The predicted classification shape-labels are then matched against stored models of pre-defined objects, in a hierarchical fashion, with the final results presented afterwar.

1) Data Augmentation

While the Hierarchical Fourier descriptors are shift, scale, and rotation invariant, they still lack the invariance to affine transformations, which create geometric distortions due to factors such as perspective irregularities, shearing, and scaling. To overcome the affine transformation limitation and improve the robustness of our database, we used a data augmentation method described in [17] to supplement our original shape dataset. We created a database of more than 200 shapes for each label, and FDs were extracted for each label and stored in a two-dimensional C++ vector, which was later serialized into an OpenCV supported .yml file. The data augmentation algorithm produces additional training database images by applying a change in aspect ratio followed by random rotation, slide, blur, and noise to the input images. Figure 9 shows the off-line training algorithm with the imbedded augmentation stage.

2) Labelled-dataset Classification

The dataset created above is essentially a collection of shape-labeled one dimensional float-valued arrays, representing the normalized Fourier coefficients. We elected to implement the SVM, kNN and distance measure based classification methods on the extracted combined features. The SVM and kNN were adapted from the OpenCV implementation. Figure 10 shows the classification stage of the recognition pipeline, allowing for different classification methods as desired by the application.

The presented results are given in Figure 11, 12. Figure 11 shows the complete pipeline recognition process, while Figure 11 shows the achieved recognition of different road signs. All presented results were achieved using distance measures, both Euclidean and Chi-squared, along with the flexibility of parameterizing the number of Fourier coefficients to be matched. The outer parent contours, which are relatively easier to identify, were matched with 20-30 coefficients. We note that the SVM and kNN methods take into account the entire feature vectors, of length 256, 512 or 1024, to build and classify the models, which at times give "over-fitted" or inaccurate results. Optimization in this process was achieved by randomly selecting a set number of feature vectors from the database, for comparisons, and implementing a multi-threaded model to compare individual child contour feature vectors.

```
Algorithm: Construct Offline labeled-database of FDs
 Data: SampleImages – Training shapes that are labeled
 Result: vector<vector<double>> databaseFDs – 2D labeled-database of normalized FDs
 begin
   for i ∈ SampleImages do
     /* Generate multiple transformations of each SampleImage at different
        scales, translations and rotations                                  */
     transformedImages ⟵ createTransformation(SampleImages[i])
   end
   for j ∈ transformedImages do
     /* Append FDs to the database along with shape labels                  */
     databaseFDs ⟵ computeFD(transformedImages[j])
   end
 end
```

Fig. 9 Off-line Training Database Algorithm

```
Algorithm: Classification of Queried FDs
 Data: vector<double> queryFD – 1D Query FDs,
       vector<vector<double>> databaseFDs – 2D labeled-database of normalized FDs
 Result: string ShapeLabel
 begin
   if (classificationMethod = SVM) then
     trainSVM(databaseFDs)
     ShapeLabel ⟵ predictSVM(databaseFDs, queryFD)
   end
   if (classificationMethod = KNN) then
     trainKNN(databaseFDs)
     ShapeLabel ⟵ predictKNN(databaseFDs, queryFD)
   end
   if (classificationMethod = DistanceMeasure) then
     /* Set the number of coefficients to be matched - Parameterized User Input(N) */
     coeffsToMatch ⟵ setCoeffsToMatch(N)
     /* Set the DB size to be matched against - Parameterized User Input(M)        */
     /* M-entries for each label will be randomly picked from the DB of FDs        */
     prunedDatabaseFDs ⟵ selectRandomly(databaseFDs, M)
     minDist ⟵ 0
     for i ∈ prunedDatabaseFDs do
       dist ⟵ calculateDistance(prunedDatabaseFDs[i], queryFD, coeffsToMatch)
       if (dist < minDist) then
         minDist ⟵ dist
         FDindex ⟵ i
       end
     end
     ShapeLabel ⟵ prunedDatabaseFDs[FDindex]
   end
 end
```

Fig. 10 Classification Algorithm

We now embed the output of the previous step i.e. the predicted shape labels of the contours, with the spatial descriptor information for each contour as calculated in the very first step. This is now a combined feature vector that will be used to match against a stored template of a known object. We will accomplish this “model matching” is a fairly straightforward and intuitive way. We start off with first matching the obtained parent shape label to the parent labels of the stored object models, once we have a match, we delve deeper into the model and start comparing the individual child shape labels, along with their spatial descriptors, with their stored template counterparts. If more than half the child labels are matched in this way, we can conclude that the parent-child contour combination belongs to the known object template. We then draw a bounding box over the queried contours along with individually predicted labels of the parent and children. The model can be modified easily and depending on the type of the problem, more tolerance can either be built into the model matching process or it can be made to follow stricter norms. In our example we use traffic signs, which are pretty robust and reliable models, consistently offering good feature to extract. This type of model matching is also invariant to a certain degree of noise and occlusion, which might make the child contours either undetectable or deformed (due to the noise).

VI. CONCLUSION

In this paper, we presented a hierarchical Fourier descriptors methods that is nested in nature. The new method captures any intrinsic spatial relationship with child contours to differentiate between shapes with similar or identical outer contours. The proposed methods was tested on a trained set of road signs that were made robust by augmenting it with additional transformed version of each image in the set. Consequently, popular classification algorithms such as SVM, kNN, and distance metrics were used to determine the shape labels. The technique performs quite well on a range of different test images, including ones from real-life scenarios, performing well even in cases where the child contours were either occluded or deformed by noise.

This work can be easily incorporated into robotic and assembly line applications, that involve detection and recognition of known objects, for quality and safety purposes. The algorithm can also easily be adapted for real time object recognition and tracking purposes. The use of the OpenCV library makes the algorithm easily adaptable to Real Time Operating Systems (RTOS) and other mobile operating platforms, like Android or Apple iOS.

The algorithm is amenable to further enhancements, for example, by incorporating into it other local and global color and interest-point based features to come up with a more detailed feature vector for the object.

REFERENCES

[1]. Dong ping & Tian (2013). A Review on Image Feature Extraction and Representation Techniques. International Journal of Multimedia and Ubiquitous Engineering, Vol. 8, No. 4, July, 2013.

[2]. T. Tuytelaars & K. Mikolajczyk (2007). Local Invariant Feature Detectors: A Survey Foundations and Trends in Computer Graphics and Vision, Vol. 3, No. 3 (2007) 177–280 2008.

[3]. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91– 110.

[4]. J.F. Canny (1986). A computational approach to edge detection. IEEE T-PAMI, 8 (6) (1986) 679–698.

[5]. Giuseppe Papari & Nicolai Petkov (2011). Edge and line oriented contour detection: State of the art. Image and Vision Computing, 29 (2011) 79–103.

[6]. S. Loncaric (1998). A survey of shape analysis techniques. Pattern Recognition, 31(8):983–1001, 1998.

[7]. D. Zhang & G. Lu (2004). Review of shape representation and description techniques. Pattern Recognition, 37(1):1 – 19, 2004.

[8]. A. Khotanzad & Y. Hong (1990). Invariant Image Recognition by Zernike Moments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(5):489–497, 1990.

[9]. El-ghazal, A., Basir, O., Belkasim, S. (2009). Farthest point distance: a new shape signature for Fourier descriptors, Signal Process., Image Commun., 2009, 24, (7), pp. 572 –586.

[10]. D. Zhang & G. Lu (2003). A comparative study of curvature scale space and Fourier descriptors for shape-based image retrieval. Visual Communication and Image Representation, vol. 14(1), 2003.

[11]. W.-T. Wong, F. Y. Shih, & J. Liu. Shape-based image retrieval using support vector machines, Fourier descriptors and self-organizing maps. Information Sciences, vol. 177, no. 8, pp. 1878–1891, 2007.

[12]. Persoon, E., Fu, K.-S (1977). Shape discrimination using Fourier descriptors. IEEE Trans. Syst. Man Cybern., 1977, 7, (3), pp. 170 –179

[13]. L.J. Latecki, R. Lakamper, & U. Eckhardt (2000). Shape Descriptors for Non-Rigid Shapes with a Single Closed Contour. IEEE Conf. Computer Vision and Pattern Recognition, pp. 424-429, 2000.

[14]. Chih-Chung Chang & Chih-Jen Lin (2005). LIBSVM, Journal of Machine Learning Research , 6, 1889-1918, 2005.

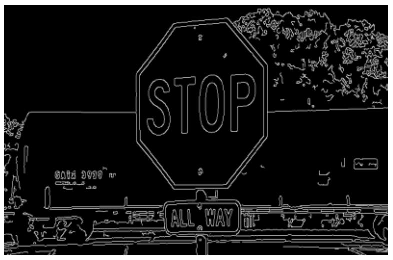
[15]. F. Larsson, M. Felsberg, & P. Forssen. Correlating Fourier descriptors of local patches for road sign recognition. IET Computer Vision, 5(4):244–254, 2011.

[16]. Nvidia (2015). NVIDIA CUDA. http://www.nvidia.com/object/cuda_home_new.html

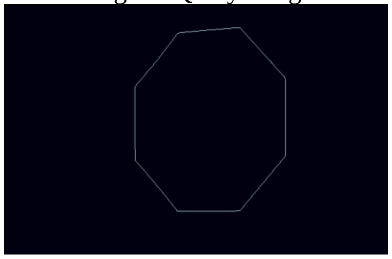
[17]. Takuya Minagawa (2014). Data Augmentation ver1.0. https://github.com/takmin/DataAugmentation .

Original Query Image

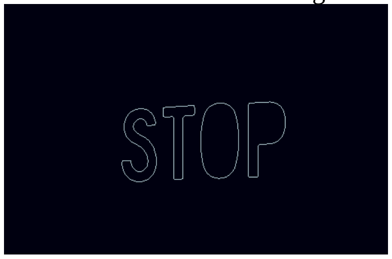
Gaussian-smoothed image



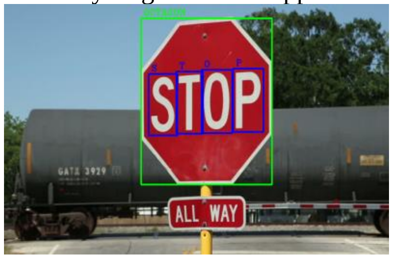
Canny Edge Detector applied



Parent contour extracted from the Canny edge map

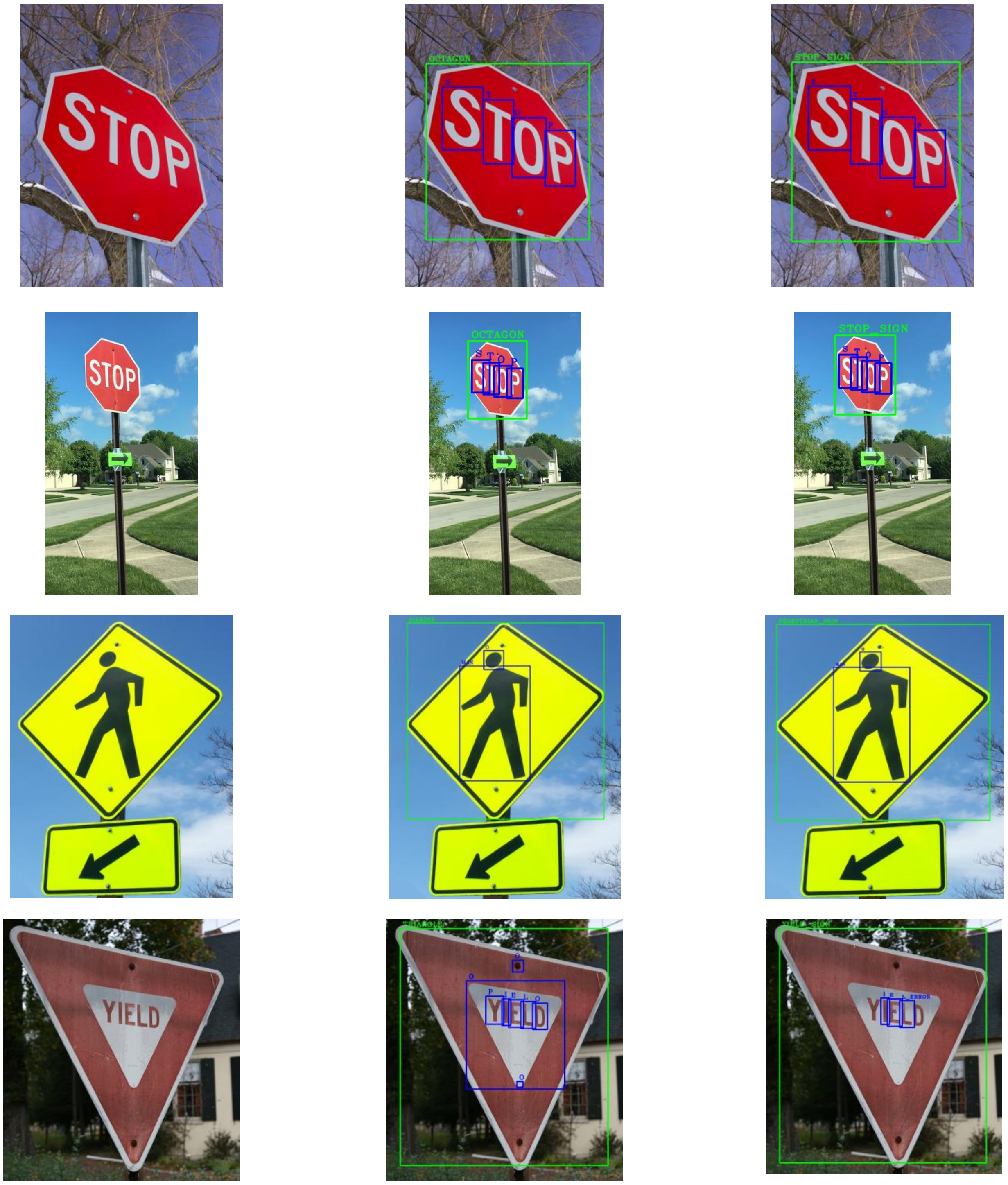


Child contours of the previously extracted parent



Classification results – labeled contours

Fig 11 Stop Sign Results



Original Contour-labels Object-model

Fig. 12 Road Signs Results