Accelerating object extraction and detection using a hierarchical approach with shape descriptors

Bassam Syed Arshad
The University of Texas Rio Grande Valley

ACCELERATING OBJECT EXTRACTION AND DETECTION USING A HIERARCHICAL APPROACH WITH SHAPE DESCRIPTORS

A Thesis

by

BASSAM SYED ARSHAD

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2016

Major Subject: Computer Science

ACCELERATING OBJECT EXTRACTION AND DETECTION USING A HIERARCHICAL APPROACH WITH SHAPE DESCRIPTORS

A Thesis
by
BASSAM SYED ARSHAD

COMMITTEE MEMBERS

Dr. Mahmoud K. Quweider
Chair of Committee

Dr. Juan Raymundo Iglesias
Committee Member

Dr. Liyu Zhang
Committee Member

August 2016

ABSTRACT

Arshad, Bassam Syed, Accelerating Object Extraction and Detection using a Hierarchical Approach with Shape Descriptors. Masters of Science (MS), August, 2016, 38 pp., 31 Figures, 33 references.

Automatic object recognition is a fundamental problem in the fields of computer vision and machine learning, that has received a lot of research attention lately. While there are different methods, that build upon various low level features to construct object models, this work explores and implements the use of closed-contours as formidable object features. A hierarchical technique is employed to extract the contours, exploiting the inherent spatial relationships between the parent and child contours of an object. Fourier Descriptors are used to effectively and invariantly describe the extracted contours. A simple hierarchical, shape label and spatial descriptor matching method is implemented, to determine the nearest object-model. Multi-threaded architecture and GPU efficient image-processing functions are adopted making the technique efficient for use in real world applications. The technique is successfully tested on common traffic signs in real world images, with overall good performance and robustness being obtained as an end result.

DEDICATION

This thesis is dedicated to my parents and siblings, for their endless support, encouragement and motivation, during the pursuit of my graduate education, despite being so far away from me.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Quweider, for having introduced me, to the interesting and compelling fields, of digital image processing and computer vision. I highly appreciate his insight, guidance, and patience, during the entire course of my master's degree, and especially his mentorship and expertise, without which most of the work presented in this thesis would not have been possible.

I would like to thank Dr. Zhang and Dr. Iglesias for having agreed to serve on my thesis committee, and supporting me all throughout my course of study. I am highly grateful to Dr. Zhang, my first graduate advisor, and the erstwhile computer science department at UTB, for having awarded me a graduate assistantship, enabling me to pursue a graduate education here, and later helping me maintain the assistantship, after the merge into UTRGV.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation

Automatic computer-based object detection and recognition is an intellectually intriguing and technologically challenging problem, in the fields of computer vision and machine learning, which finds a variety of applications in, advanced military systems, state of the art surveillance methods for law enforcement, fundamental robotics-related tasks, assembly line manufacturing and quality control systems, and critical medical imaging software's used in the diagnosis and treatment of many diseases. The problem proves to be particularly gripping, to an investigator, given the fact that it is a very intuitive activity for us humans, whereas it translates into a complex and tedious task for machines. This particular area has generated a lot of research interest in the last couple of decades, going hand-in-hand with advances in electronics, imaging modalities, computing hardware, as well as, distributed and parallel computing. Various techniques and approaches have been proposed in order to tackle the problem, which at times can be based upon fundamentally different building-blocks.

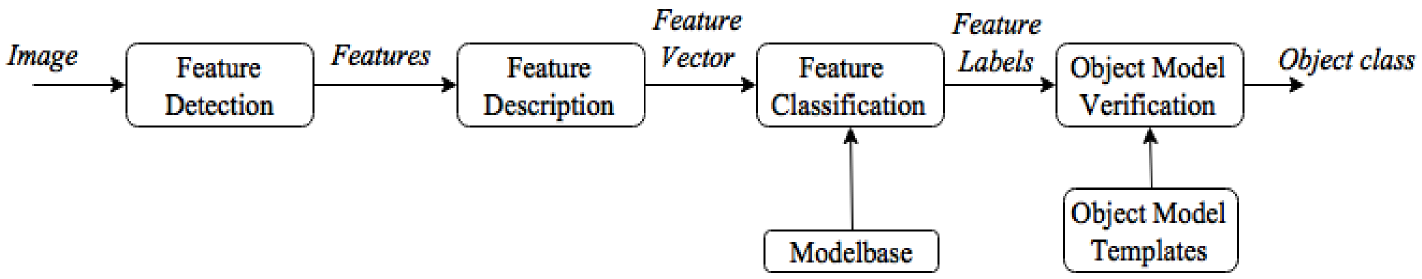


Fig 1.1: An object recognition pipeline

The object recognition problem can be effectively divided into three separate sub problems: Feature Detection, Feature Description and Feature Classification. The three sub problems are described briefly, in context of digital image processing and computer vision applications, below:

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. There are different types of features that can be defined in images, and we go over them in the next chapter.

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.

After an initial review of the many proven methods, that aim to solve the object recognition problem, it is quite evident that the primary step of feature detection plays a decisive role in determining the success of the process. In the next chapter we will go over the various types of features that can be used to quantify an object in an image or a scene.

Contours are reliable and highly descriptive features, that are derived from image edge maps. While objects have previously been represented by their outer contours/silhouettes, inner contours which can formulate a hierarchical representation of an object have not been explored. In this work we particularly focus on the hierarchical relationships between the outer contour/shape of an object and the inner contours contained by it. These parent-child contour relationships can later be successfully described and built into a model to uniquely represent various objects.

For the technique to find real world practical applications, it requires the description of the detected features, in our case contours from the object, to be invariant to changes in translation, rotation and scale. In real world images and scenes, objects can appear at different distances, tilts and perspectives. While to us human beings, thanks to our well-developed perceptive abilities, that evolve incrementally with age and experience, we can distinguish objects successfully in different scenarios. In order for a machine to be able to make these type of distinctions, algorithms and techniques are devised that can successfully model object features into formats that offset the effects of these transformations. The Fourier Descriptors (FDs) is one such technique that addresses the above challenges.

We select some of the most common traffic signs, as a test case, for our object recognition pipeline. Training images, were augmented, by rotating and scaling them to different levels, in order to achieve added robustness during the classification step. Different machine learning based classification approaches were implemented in the software, and we discuss the pros and cons for them, in context of our feature description method.

The end goal of this work was to formulate an object recognition technique, based on the hierarchical representation of an object's contours.

MATLAB is one of the most commonly used tool in building and prototyping, image processing and computer vision applications. While MATLAB provides greater ease of use and quick command-line access to various in-built utilities, getting optimized multi-platform executables and binaries is not always feasible. We implemented this work in the C++ programming language, taking advantage of, the in-built STL libraries, and freedom of defining complex data structures. The exploration and use of the C++ OpenCV library was crucial in achieving the desired results, especially the use of GPU-intensive functions implemented in the library. OpenCV provides functionalities to implement almost all the concepts covered in this work, right from the basic data structures to store images, functions to extract contours and perform DFT's, to machine learning algorithms like the kNN, SVM and ANN, and GPU optimized routines for low-level computationally expensive tasks.

1.2 Problem Statement

Algorithmically, using the divide and conquer approach, our problem can be broken into:

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.

1.3 Contribution

This work proposes a new hierarchical technique for contour-based feature extraction, in which an object is identified by a set of contours representing its overall structure. A custom algorithm was developed in order to extract these hierarchical parent-child contours and discard false child contours. To implement the new model, a custom tree-like data-structure was created to store and retrieve the object features, which facilitated in efficient matching against stored object models. While the proposed technique ignores color and texture features (which can be integrated in later stages), it does add another level of information in the form of smaller nested contours, which make the method more robust and discriminant over existing ones. The built in GPU-functionality, that comes integrated into many standard OpenCV algorithms, was used to gain a considerable speed-up in the entire process, especially on computer systems with powerful graphic cards. In order to test and prove the validity of the proposed technique, common traffic signs were selected as sample objects. The results of the tests are later presented in this work, confirming the robustness and accuracy of the technique. The C++ and OpenCV platform makes the application, in its current form, ready for deployment on RTOS and mobile platforms. Classes and methods have been independently defined and implemented for the three different stages of the technique, namely: Feature Extraction, Feature Description and Feature Classification, facilitating prospective users to define their own objects and models. Finally, the source code for this work is being made available publicly for further enhancements and research.

CHAPTER II

BACKGROUND

This chapter describes some of the prevalent techniques in the areas of feature detection/extraction, feature description and feature classification, in context of the object recognition problem, along with a brief introduction to the OpenCV library. We also start building a base for the next chapter and will focus on the techniques that we shall be employing.

2.1 Feature Detection

In the context of image processing and more specifically the problem of object detection, a feature can be defined as a point or region of interest, that can be used to uniquely describe the object. A "good feature" appears consistently in images containing the object, is insensitive to noise, and is invariant to transformations in scale, rotation and shift, as well as affine transformations, of the object.

Some of the most popular features used today include color, shape, corners/interest-points and texture, as reviewed by Tian [1]. 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 [2] and edges have been vastly used. 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 [4] is probably the best known and widely used technique, in this category. See [3] for a good survey on local invariant feature detectors.

Edges are some of the lowest-level features that can be extracted from an image. Edges originate in areas of the image, with significant gray-level intensity contrast. Typically edge detection involves some form of smoothing, followed by taking a derivative or the gradient of the image. An in-depth review and comparison of various edge detection methods is provided in [5]. One of the most widely used edge detectors is the Canny edge detector [6,7], which is designed as an “optimal” edge detector, that is capable of extracting edges at varying scale-levels, by adjusting the Gaussian sigma value. It basically extracts the gradient of the image, using a Gaussian derivative and then suppresses the non-maximal responses using two given gray-level intensity thresholds. It is also important to note here that not all edges might be extracted for a given sigma (standard deviation) value, the higher sigma value corresponds to a larger Gaussian blur, hence the finer edges are lost. A lower sigma yields more detail, but at the same time, we deal have to deal with more noise. A Gaussian pyramidal approach is often used in conjunction with edge detection techniques and is referred to as multi-resolution edge detection [31], which basically extracts edges at different scales, corresponding to a Gaussian sigma value and varying-resolution of the gray-image, at each level of the pyramidal data-structure.

Object contours can be defined as the distinctive set of edge-boundaries that differentiate the object from its surrounding scene. Contours are of high significance, and can describe important shape attributes of the object. Various techniques and state of the art approaches for contour detection, are discussed in-depth in [8]. It is important to note that the contour detection problem is significantly more complex than basic edge detection.

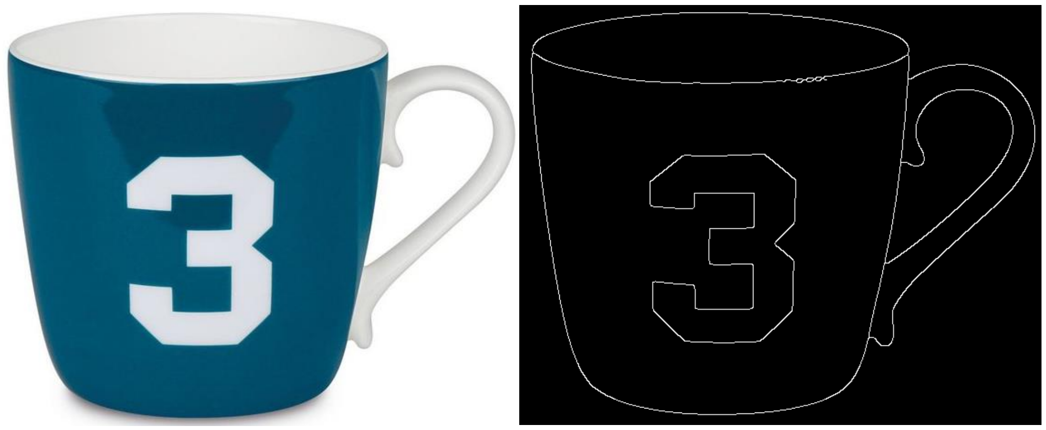


Fig.2.1 An object and its corresponding canny edge map image.

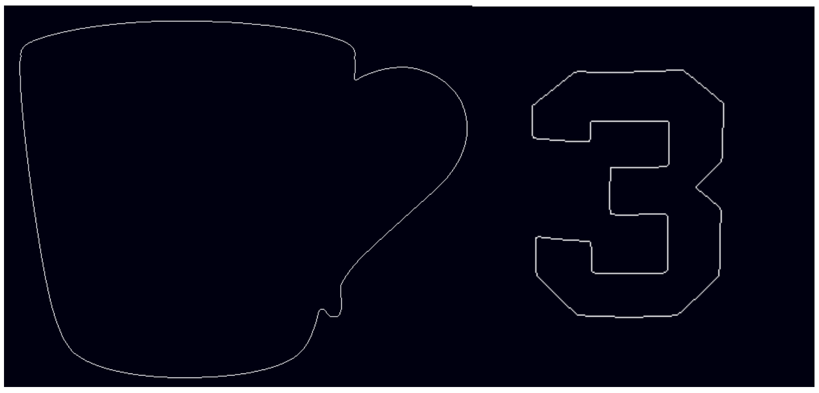


Fig.2.2 Parent and Child contours of the object.

A contour detector should be able to distinguish between texture edges, region boundaries and object contours. We achieve this by applying a host of pre-processing operations on the image first: smoothing the image using a Gaussian filter - this enhances the edge detectors response while distinguishing between region boundaries, applying texture smoothing filters - bilateral filter or a mean-shift filter - to suppress texture edges, finally some sort of morphological operations - dilation or closing might be required to obtain good closed object contours.

Having obtained well-defined object contours we can now start building on top of these shape features, which can then be described using different techniques.

2.2 Feature Description

Having obtained the features, that are discussed in the previous section, we will now focus on the various feature descriptors, laying particular emphasis on feature descriptors for closed shapes or contours.

A good feature descriptor is a one that can embed/encode a high-level of information/characteristics about the feature, normalizing the effects of shift, scale, rotation and affine-transformations, into a low-dimensionality/lightweight feature vector.

Quite intuitively for color features, RGB/HSV histograms are popular feature descriptors. In case of the RGB histogram we obtain a three-dimensional feature vector, with 256 bins for each channel.

Interest points based features, can also be uniquely described in terms of their local neighborhood. A simple but naive approach would be to extract a gray-level intensity histogram, around a 3x3 or 5x5 neighborhood centered over the interest point. More refined and complex approaches exist that take into account the spatial characteristics of the neighborhood around the interest point. The SIFT feature descriptor, constructs a gradient orientation histogram over a 16x16 window centered over the interest point. The resultant feature vector is further normalized to make it robust to photometric variations and to reduce the effects of contrast/brightness variations [4]. These type of spatial histogram based feature descriptors have been proven to be insensitive to deformations in the object, and invariant to changes in scale and rotation.

Shape descriptors can be broadly classified into two categories *region based* and *contour/boundary based*. Both these categories can be further subdivided into *local* and *global* methods. Global methods aim at describing the entire shape as a whole, while local methods partition the shape into sub-parts which are then described individually and provide robustness to partial object occlusion. [9,10], cover shape based feature description and matching in great detail.

Region based methods are targeted towards generating a more global description of the entire object, including areas encompassed within the object. Some of the popular region based approaches focus on the calculation of image moments, like the Hu’s Moments [11] or the Zernike Moments [12].

Contour or boundary based methods concentrate on the description of each given contour, individually. These methods fall into the category of *local* methods, and have their own advantages, like better invariance to occlusion and an ability to define global abstraction for the object model. Individually invariant contour descriptions, can be later used, along with the spatial information of the individual contours, to build more complex models of objects. Nested hierarchies can be defined amongst these contours and models can be built accordingly, which can be later matched, based on the level of abstraction defined by the user.

Some of the popular methods in practice for contour description are chain codes, curvature scale space(CSS), shape context and the Fourier descriptors, FD’s.

Chain codes are one of the oldest techniques for contour description, originally proposed by Freeman in the 1960s [13,14]. For the calculation of chain codes, given a connected contour, a 4-neighbor or 8-neighbor pixel connectivity is defined. A number/code is assigned, indicating the direction of the movement between each adjacent pixel. This is continued until the starting pixel is reached again. Start-point invariant codes can easily be extracted, which makes this technique rotation and shift invariant, but such representations are quite susceptible to noise and achieving scale-invariance is also a problem.

Shape context is also a popular technique among contour based feature descriptors [15]. The contour is first sampled for a set number of points, and then a log-polar histogram of edge energy is derived around the sampled points. Individual histograms can then be matched by using chi-square test statistics. Shape context is moderately invariant to small non-rigid deformations.

The Fourier descriptors (FDs) [16, 17] is a classic and valid method for contour description. The shape description and classification using FDs are simple to compute, robust to noise. The key concept is application of the Fourier transform to a periodic, continuous and closed representation of the contour, which will then give us a shape descriptor in the frequency domain. The low-frequency components of the Fourier spectrum correspond to the general shape of the contour while the finer details are described in the high frequency components. This property is especially useful in two ways. Firstly, while matching two different shapes, a comparison between a few low-frequency components can give us a good similarity distance, between the two shapes. Secondly, the effects of noise can be nulled to a certain extent, while matching similar shapes i.e. effects of minor deformities are generally represented by the higher frequency components.

The first step here is to resample the contour to a fixed number of points, this is also known as the 1D parameterization of the contour. This can be achieved in different ways, curvature/perimeter based, distance to the contour centroid, angular steps etc. In the second step the resampled contour coordinates are represented as complex numbers, the x-coordinate usually corresponding to the real part of the complex number and the y--coordinate as the imaginary part. Various techniques for the calculation and representation of FDs are explained and reviewed by Nixon *et. al.* in [19].

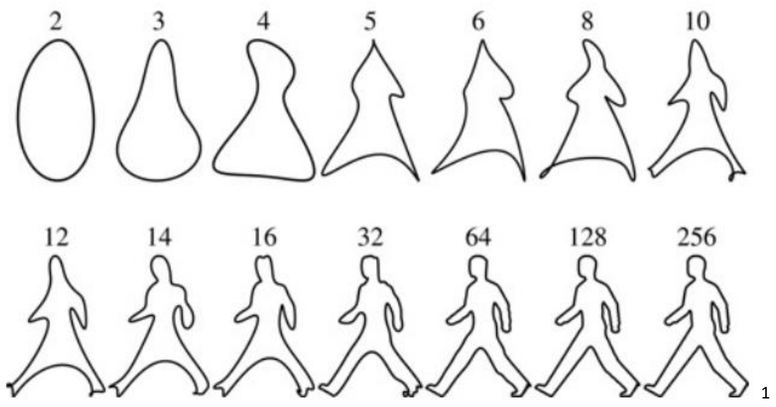
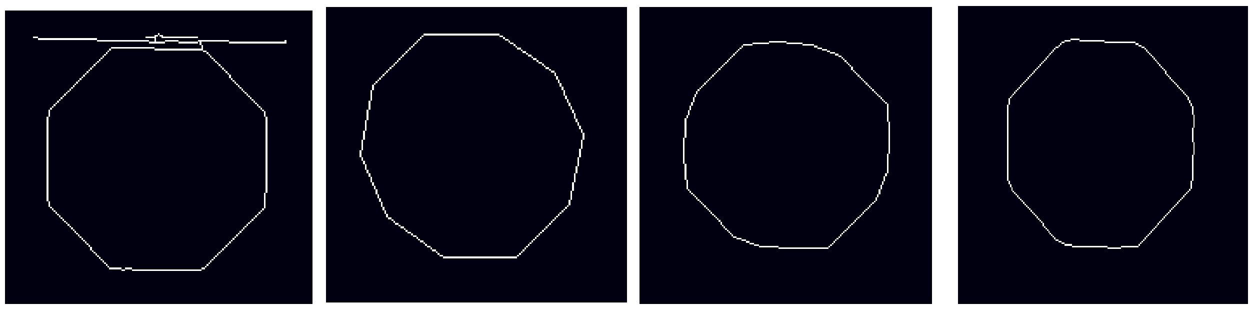


Fig.2.3 Shows the reconstruction of a shape using increasing number of Fourier coefficients[1]

We will not divulge into the details behind the mathematics and concepts regarding the Fourier series and Fourier transform, which have been covered in great depth, in the standard text by Bracewell [20], we will discuss applications, analysis and properties of the Fourier spectrum, derived from the resampled contours points.



Original Contour n=10 n=20 n=50

Fig 2.4: Smoothing effect of FDs-based reconstruction of a noisy contour (Contour resampled over 512 points, n→ number of coeffs used for reconstruction).

[1] F. Larsson, M. Felsberg, and P.-E. Forss´en. "Correlating Fourier descriptors of local patches for road sign recognition". IET Computer Vision, 5(4):244–254, 2011.

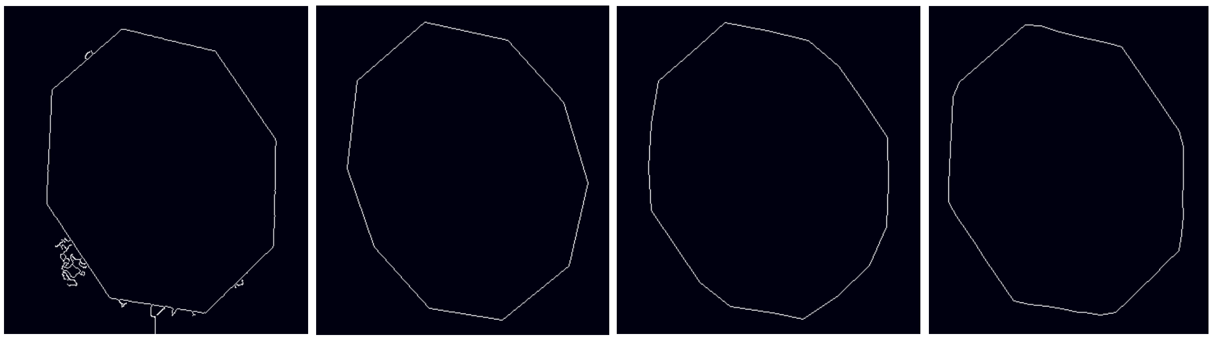


Fig 2.5: Smoothing effect of FDs-based reconstruction of a noisy contour (Contour resampled over 512 points, n→ number of coeffs used for reconstruction).

As described in the work of D. Zhang *et.al.* [21], the Fourier spectrum can be easily made invariant to changes in scale, translation and rotation. Scale invariance is obtained by simply dividing the entire spectrum by the dc-component or zero-frequency term, present in the Fourier transform. By ignoring the phase information and using only the magnitude of the coefficients, the descriptor achieves invariance to shift and rotation.

The one-dimensional, fixed length, Fourier descriptor feature vector, is lightweight and computationally inexpensive, when it comes to feature matching and classification.

2.3 Feature Classification

Having obtained a feature vector (FV), whether it be a one-dimensional FD, 128-dimensional SIFT descriptor or a 3-dimensional RGB/HSV color histogram, there exist multiple techniques, that can be used to find the closest match for a given query FV, against a database of FV's.

Some of the popular techniques, again in context of the object recognition problem are: Support Vector Machines (SVM), k-Nearest Neighbor approaches (kNN), Artificial Neural Networks(ANNs), or various distance measures (Euclidean, Chi-Squared, Cosine-similarity etc.). We will focus on some techniques used in classifying or matching 1D FDs.

FV obtained from the FDs, in their basic form (normalized Fourier coefficients), can be fed into a SVM, kNN or an ANN classification algorithm. SVM have been used successfully to classify shapes, using FDs in [22], the paper analyzes the implementation technique over the standardized MPEG-7 test database of various object shapes [24]. Similarly, the kNN method has also been used quite frequently in order to classify FDs, an application of the same to the character-recognition problem is presented in [25].

One of the limitations of the SVM is that they do not natively support multi-label or multi-class classification, as the classification is performed in a one vs all fashion. However, there exist implementations of the SVM algorithm that try to circumvent this limitation, one such is the LIBSVM [26].

When trying to classify shapes using FDs one common issue is the size of the FV, that we would like to retain and is relevant for matching problem. As we have discussed earlier, based on how the DFT is distributed, different term in the Fourier spectrum correspond to different characteristics of the object's shape. This property is of more significance especially when taking into account, the presence of noise in the queried shape. By comparing just a few low-frequency coefficients in the FV we can ignore the effects of noise and incorporate robustness into the system.

A faster and easily parameterized approach is the use of distance measures [21], between the FDs. The number of coefficients to be matched can be parameterized depending on the scope of the problem. Moreover, as shown in [23,27], instead of discarding the phase information in the FDs, by taking the absolute value of the complex DFT output, to obtain rotation and shift invariance, the same can be obtained by estimating the least-squared error for rotation and shift.

2.4 OpenCV

OpenCV is an open source library consisting of standard and start of the art implementation of computer vision algorithms. Originally started as a research project at Intel in 1999 with the goal of mainstreaming computer vision application development, it is currently managed by the non-profit organization OpenCV.org [28]. OpenCV provides for its own efficient data structures to store and manipulate images, and functions that perform complex linear algebra operation on the same. Apart from optimized function for matrix algebra, the library also provides standard frequency-domain signal processing functions, like the DFT (Discrete Fourier Transform), and in-house implementations of machine learning algorithms, like the SVM, kNN and ANN. Recently, and more importantly, the library has been implementing GPU-intensive variants of standard, as well as complex, computer vision algorithms. These variants work in partnership with the NVIDIA CUDA platform [29], for now, and a considerable performance boost can be obtained over several just CPU-intensive algorithms. A detailed overview of the OpenCV library has been presented in [30].

CHAPTER III

OVERVIEW OF THE PROPOSED TECHNIQUE

A method will be 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.

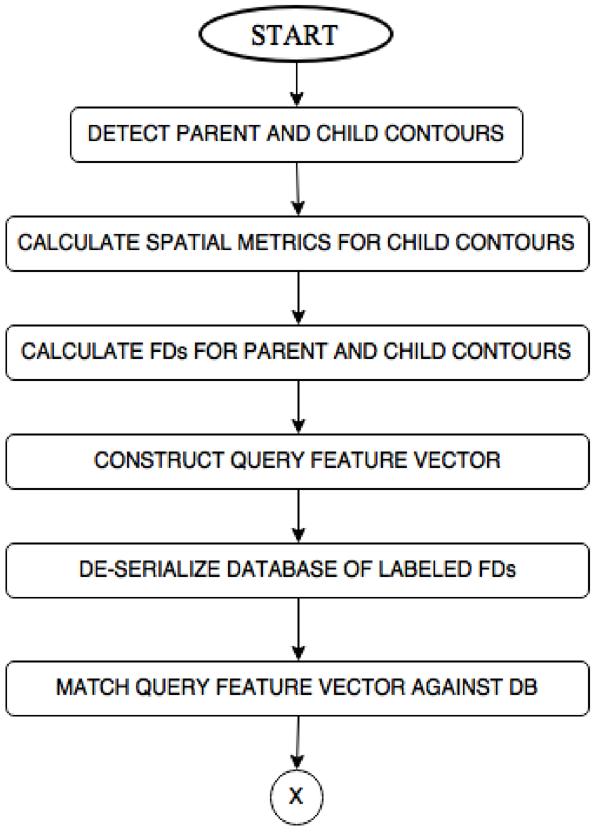

START
DETECT PARENT AND CHILD CONTOURS
CALCULATE SPATIAL METRICS FOR CHILD CONTOURS
CALCULATE FDs FOR PARENT AND CHILD CONTOURS
CONSTRUCT QUERY FEATURE VECTOR
DE-SERIALIZE DATABASE OF LABELED FDs
MATCH QUERY FEATURE VECTOR AGAINST DB
X

Fig 3.1: Flowchart of proposed object recognition technique (part 1)

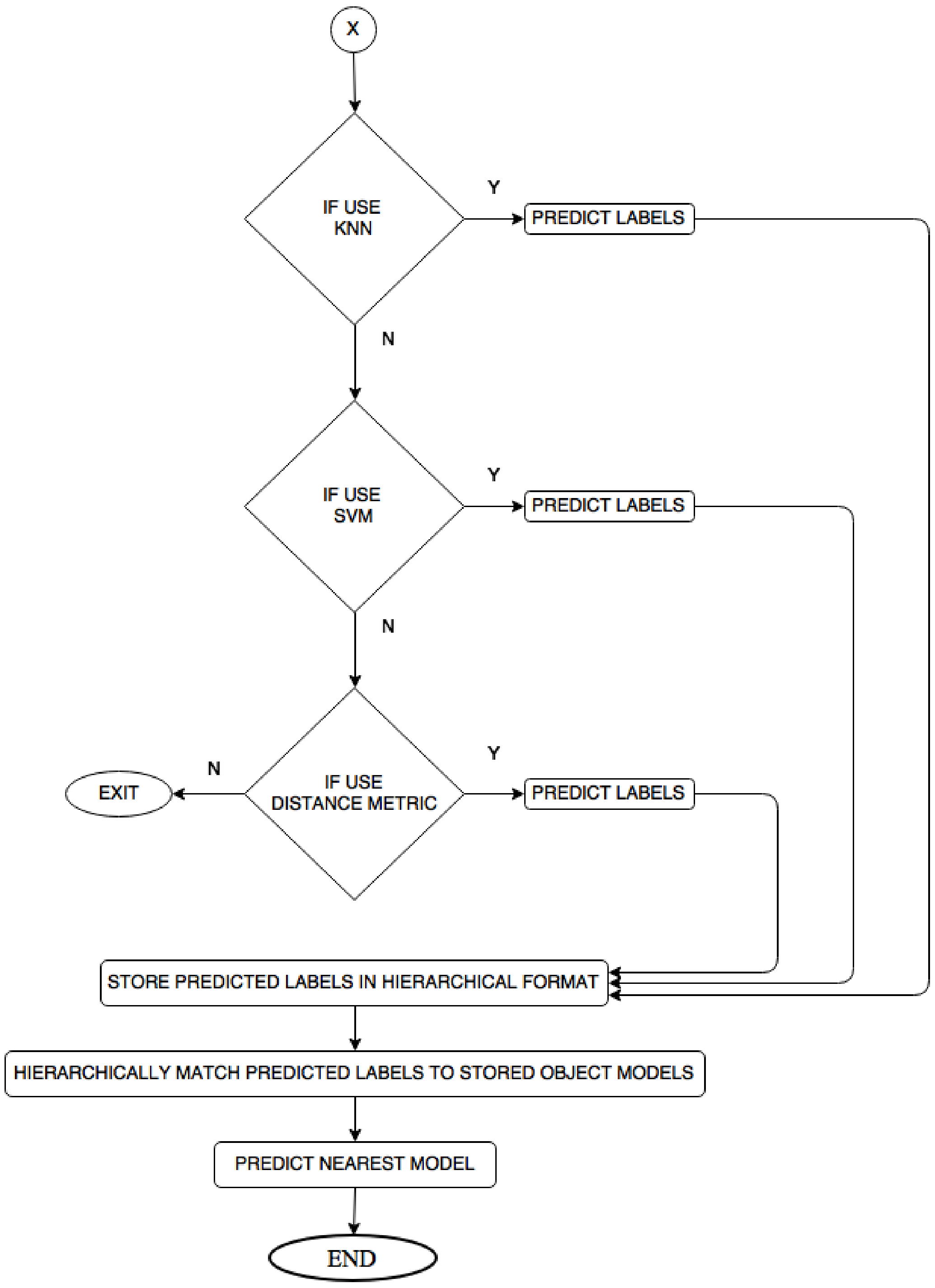


Fig 3.2: Flowchart of proposed object recognition technique (part 2)

CHAPTER IV

METHODOLOGY AND RESULTS

In the first step we will design a contour extraction technique that will look for candidate parent contours. Once a closed contour, above a certain area threshold is detected, we look for the closed child contours it encompasses. Here again we set an area threshold, while looking for candidate child contours, with respect to the parent contour, this is done in order to discard small noisy child contours. We also calculate spatial descriptors for the child contours with respect to the parent contours. We define three metrics here: child-parent contour centroid distance (normalized by parent perimeter), child contour aspect ratio and child contour orientation. A hierarchical data structure is used to store the parent contours and their respective children, along with the spatial descriptors for each of them.

```
Algorithm: Extract Parent and Child Contours
  Data: QueryImage
  Result: tuple <parentContour, childContours, childSpatialDescriptor> parentChildContours
  begin
    contours ⟵ findContours(QueryImage)
    for $i \in$ contours do
      if candidateParentContour(contours(x)) then
        parentContour ⟵ contours(x)
      end
    end
    allChildContours ⟵ getAllChildContours(parentContour)
    for $j \in$ 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 4.1: Algorithm to extract parent and child contours

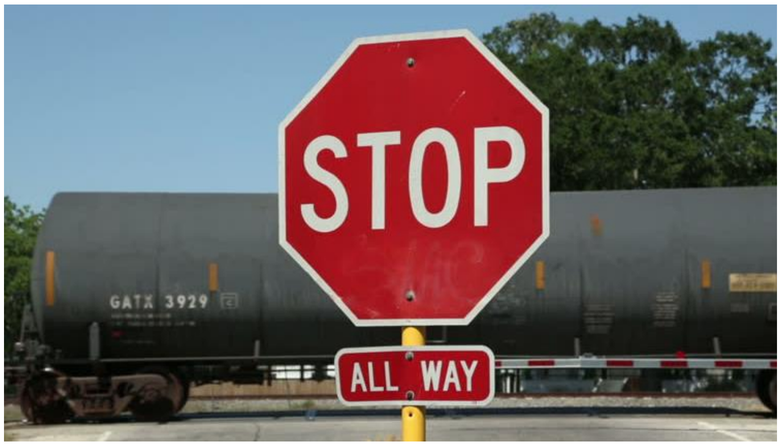


Fig 4.2: Query Image

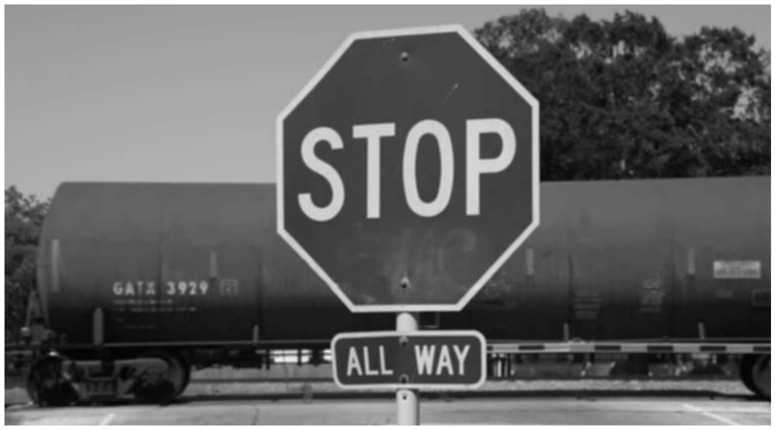


Fig 4.3: Smoothed using Gaussian filter (σ = 1.5)

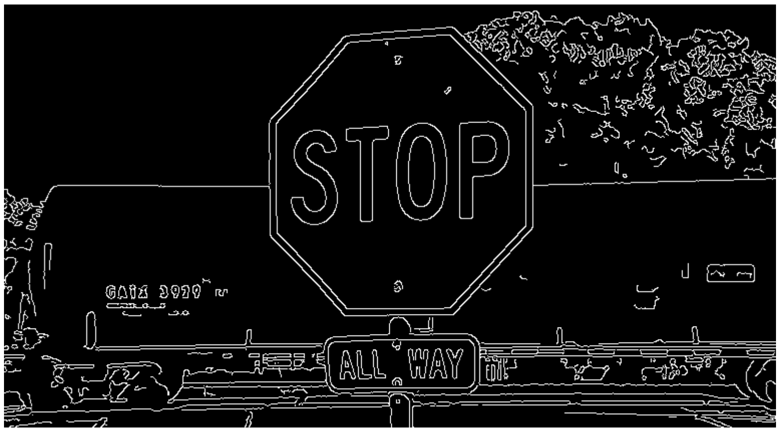


Fig 4.4: Canny Edge Detector applied (kernel size =3)

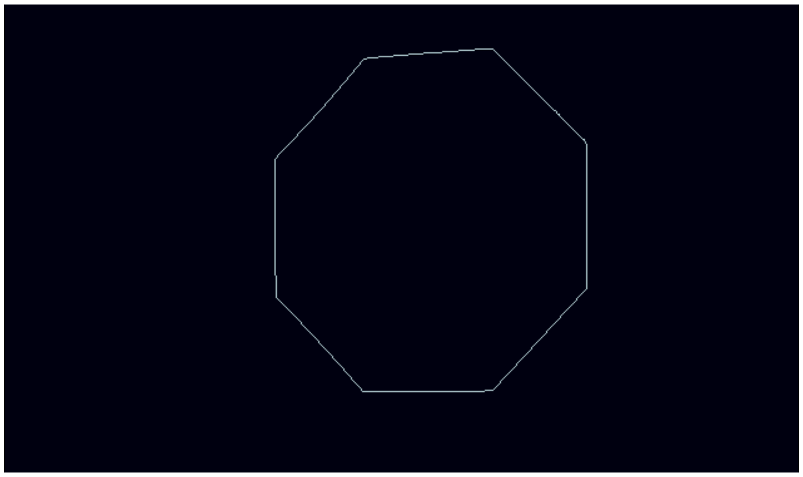


Fig 4.5: Parent Contour

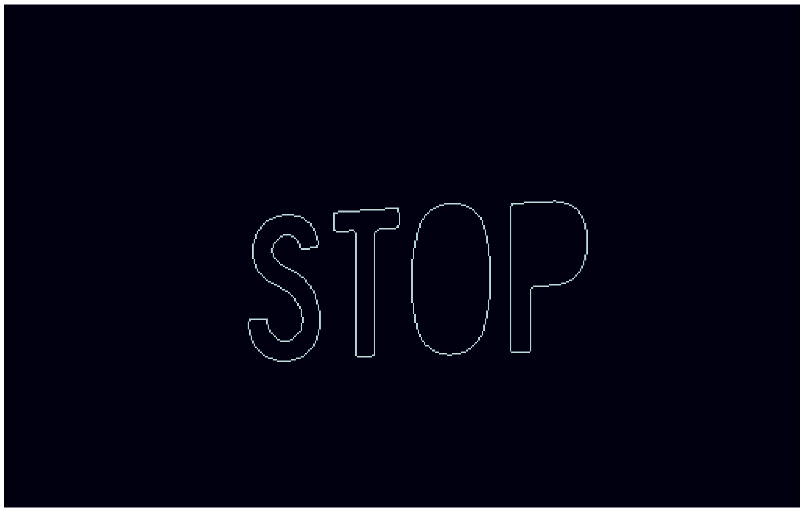


Fig 4.6: Child Contours

In the second step, we calculate the FDs for the parent and child shapes extracted. This is done as shown in [23], wherein, we will first resample the contours to either 256, 512 or 1024 points, then convert each point into a complex coordinate and finally calculate the DFT. The reason behind the selection of these numbers, is to facilitate the implementation of the optimized radix-2 FFT algorithm. The Fourier coefficients obtained are divided by the dc-component of the spectrum, which the OpenCV function places at index 0 of the one-dimensional vector. Furthermore, we take an absolute value of the resultant complex Fourier coefficients. As explained earlier, the above operations will make the Fourier coefficients invariant to changes in scale, translation and rotation.

```
Algorithm: Compute Fourier Descriptors
  Data: contour, noOfPoints
  Result: vector<double> normalizedFD  - 1D array of normalized FDs
  begin
      $contour \longleftarrow ReSampleContour(contour, noOfPoints)$
      for $i \in contours$ do
          /* Converting to complex coordinates                                   */
          $complexContour \longleftarrow contour[i].x + j(contour[i].y)$
      end
      $FD \longleftarrow DFT(complexContour)$
      /* Making the FDs Scale Invariant - dividing by the dc-component           */
      $FD \longleftarrow FD/FD[0]$
      /* Making the FDs Shift and Rotation Invariant - Taking the absolute value */
      $normalizedFD \longleftarrow abs(FD)$
  end
```

Fig 4.7: Algorithm to calculate Normalized Fourier Descriptors

The successful working of any classification technique or model is directly dependent on the quality of features used to train it. The FDs are in themselves a scale, translation and rotation invariant technique to represent contours, but in order to obtain added invariance to affine transformations and improve the accuracy of our method, in order to be applied to real world test cases, we used a data augmentation method [32] 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.

```
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 $i \in SampleImages$ do
          /* Generate multiple transformations of each SampleImage at different
             scales, translations and rotations                                   */
          $transformedImages \longleftarrow createTransformation(SampleImages[i])$
      end
      for $j \in transformedImages$ do
          /* Append FDs to the database along with shape labels                   */
          $databaseFDs \longleftarrow computeFD(transformedImages[j])$
      end
  end
```

Fig 4.8: Algorithm to construct offline database of FDs

The dataset we have created is essentially a collection of shape-labeled one dimensional float-valued arrays, representing the normalized Fourier coefficients. There are many machine-learning based methods that can be implemented to classify this labeled-dataset. We implemented the SVM, kNN and distance measure based methods. The SVM and kNN were adapted from the OpenCV implementation. The best classification results were achieved while using distance measures, both Euclidean and Chi-squared, along with the flexibility of parameterizing the number of Fourier coefficients to be matched. For example, at the parent level, we do not look for a very exact match, and it was observed that by matching 20-30 coefficients, a general shape match can be accomplished. On the other hand, 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 will 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: 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
        $trainSVM(databaseFDs)$
        $ShapeLabel \longleftarrow predictSVM(databaseFDs, queryFD)$
    end
    if ($classificationMethod = KNN$) then
        $trainKNN(databaseFDs)$
        $ShapeLabel \longleftarrow predictKNN(databaseFDs, queryFD)$
    end
    if ($classificationMethod = DistanceMeasure$) then
        /* Set the number of coefficients to be matched - Parameterized User Input(N) */
        $coeffsToMatch \longleftarrow setCoeffsToMatch(N)$
        /* 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       */
        $prunedDatabaseFDs \longleftarrow selectRandomly(databaseFDs, M)$
        $minDist \longleftarrow 0$
        for $i \in prunedDatabaseFDs$ do
            $dist \longleftarrow calculateDistance(prunedDatabaseFDs[i], queryFD, coeffsToMatch)$
            if ($dist < minDist$) then
                $minDist \longleftarrow dist$
                $FDindex \longleftarrow i$
            end
        end
        $ShapeLabel \longleftarrow prunedDatabaseFDs[FDindex]$
    end
end
```

Fig 4.9: Algorithm for classification of queried FDs

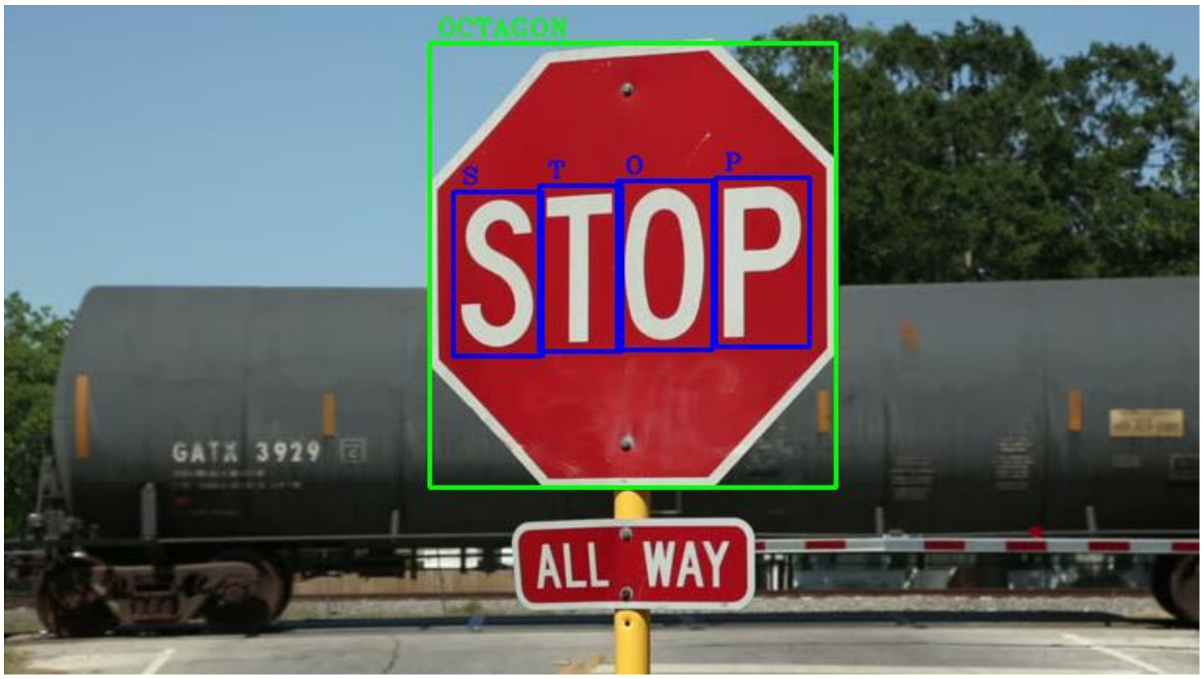


Fig 4.10: Classification results – labeled contours

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 will 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).

```
Algorithm: Object Model Matching
 Data: vector<pair<$string, double$>> queryShapeLabels – 1D vector of shape label and spatial descriptor
          pairs, from query image.
          vector<pair<$string, double$>> modelShapeLabels – 1D vector of shape label and spatial descriptor
          pairs, from stored model.
 Result: $string$ ObjectModel and $BoundingBox$ ObjectBox
 begin
     /* The parent shape is stored at index 0, in both the vectors.                          */
     /* We will match the query and stored-object models in a hierarchical order i.e.
        first parent shapes then corresponding child shapes.                                 */
     if $(queryShapeLabels[0] = modelShapeLabels[0])$ then
         /* Starting for loop with i=1, in order to access child shapes.                     */
         $i \longleftarrow 1$
         for $i \in queryShapeLabels$ do
             /* Read the two-value pairs into local variables.                               */
             $queryChildShape \longleftarrow queryShapeLabels[i].first$
             $modelChildShape \longleftarrow modelShapeLabels[i].first$
             $queryChildSpatialDesc \longleftarrow queryShapeLabels[i].second$
             $modelChildSpatialDesc \longleftarrow modelShapeLabels[i].second$
             if $((queryChildShape = modelChildShape)$ AND
              $(queryChildSpatialDesc = modelChildSpatialDesc))$ then
                 $shapesMatchedModel{+}{+}$
             end
         end
         /* We look for a match between, at least, half the query child shapes and the model
            child shapes.                                                                    */
         if $(shapesMatchedModel >= ((modelShapeLabels.size() - 1)/2))$ then
             $ObjectModel \longleftarrow modelShapeLabels.name()$
             $ObjectBox \longleftarrow highlightAndBoundObject(queryShapeLabels)$
         end
     else
         /* In case parent shapes do not match.                                              */
         getNextModel()
     end
 end
```

Fig 4.11: Algorithm to match query-labels against stored object models.

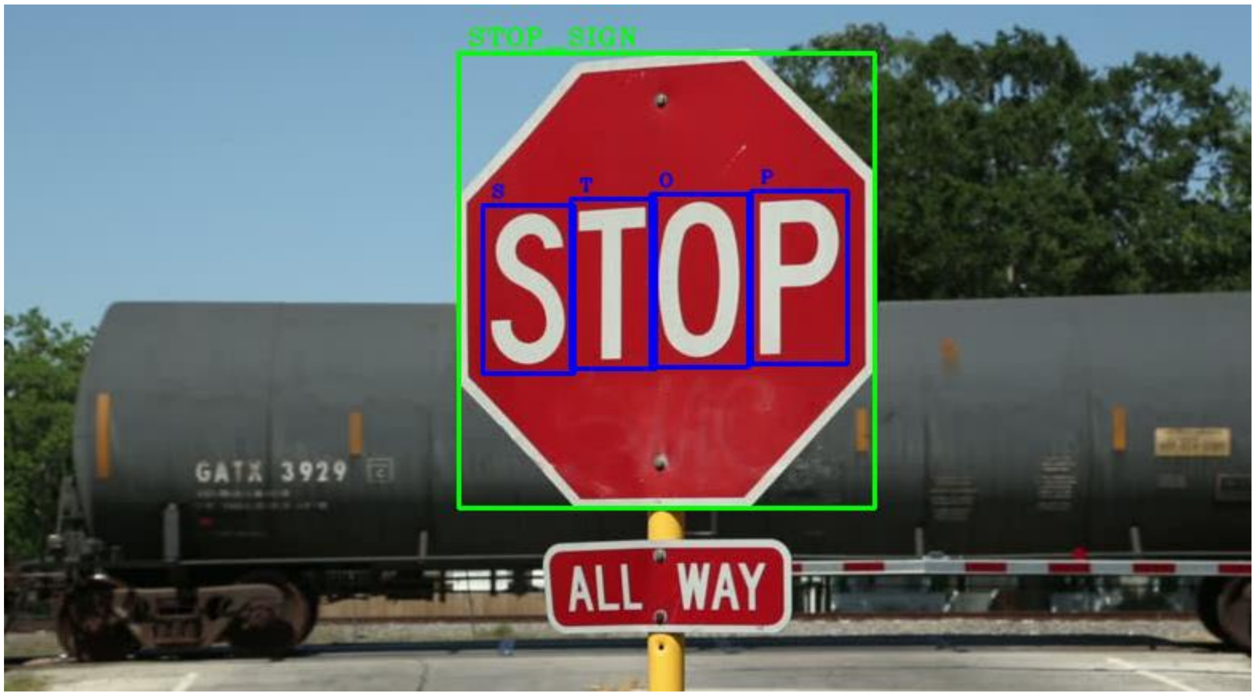
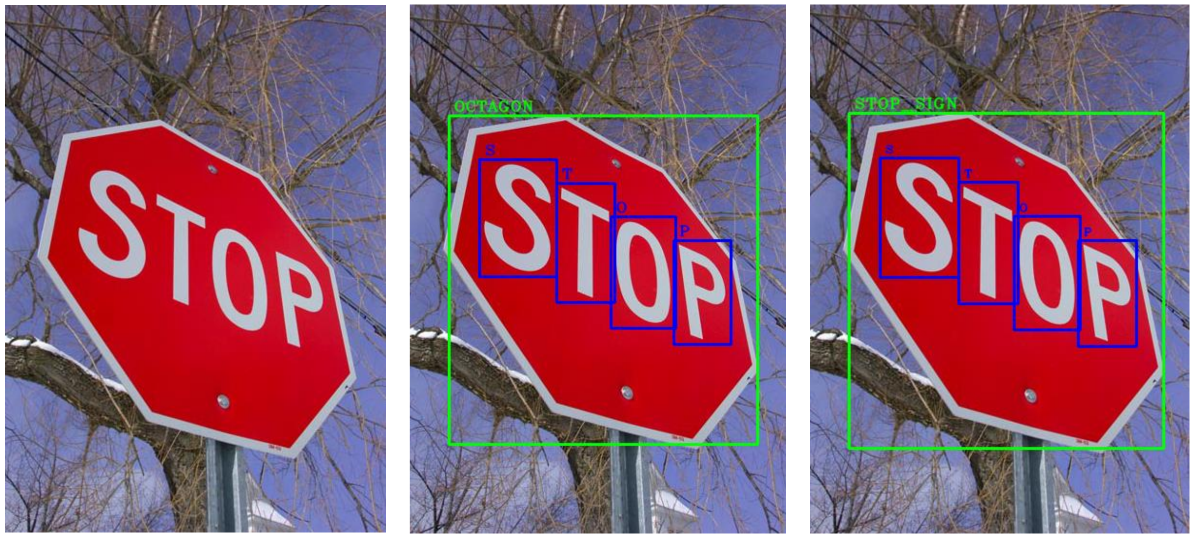


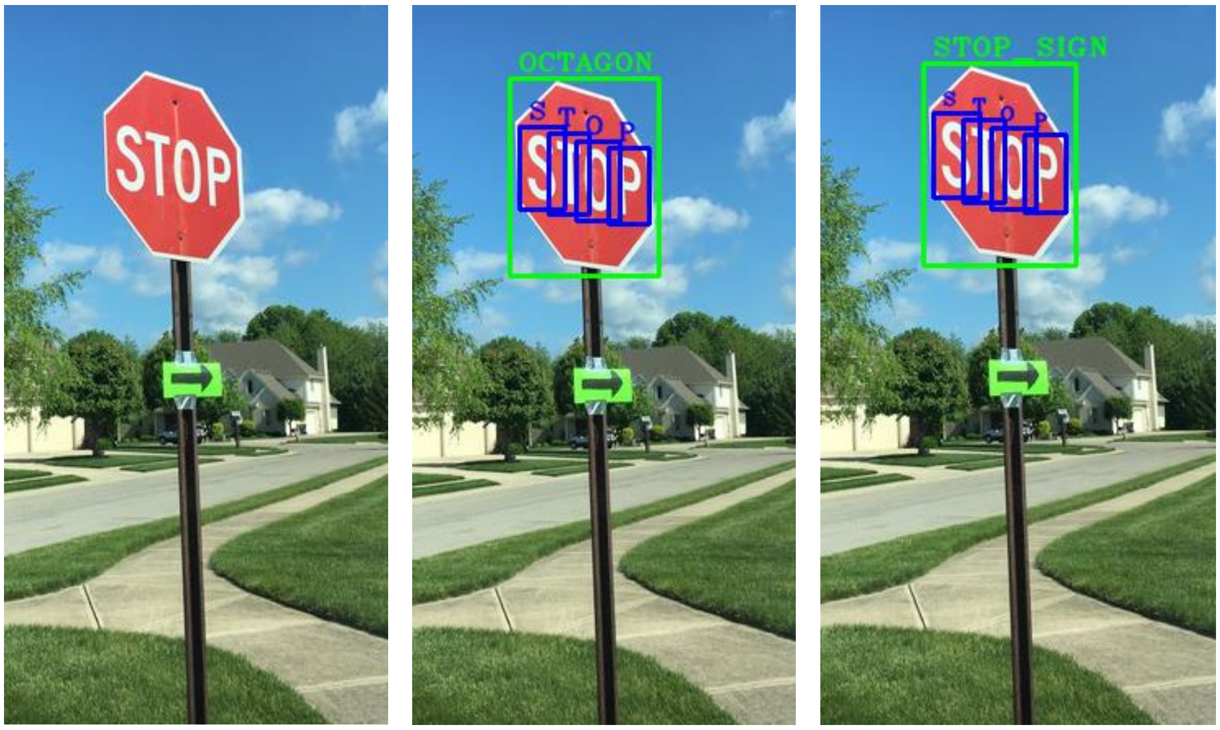
Fig 4.12: Result - Nearest Object Model after label-matching

Below are the results from some of the tests that were conducted:



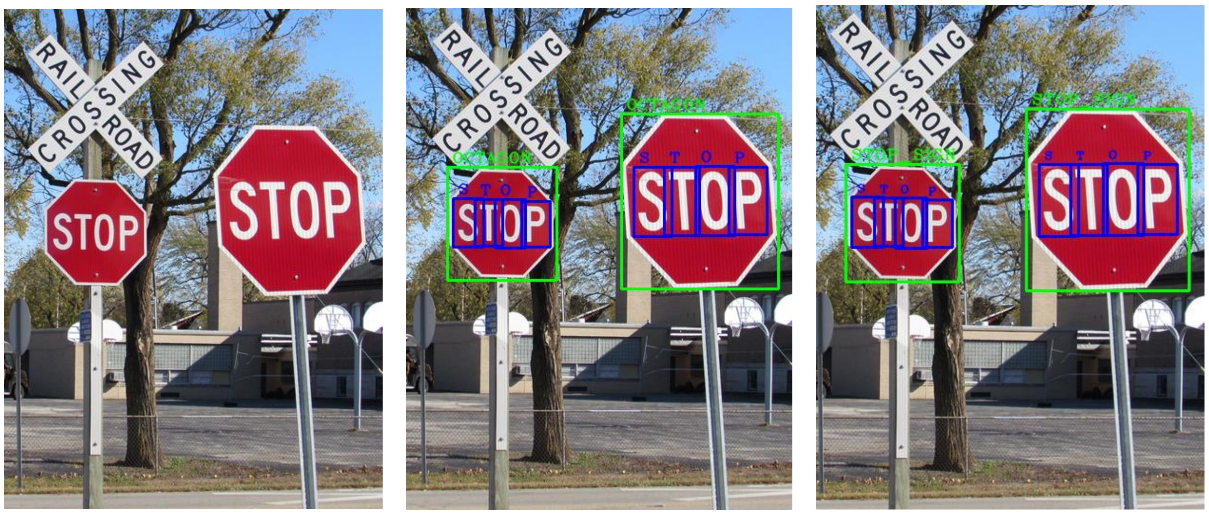
Original Contour-labels Object-model

Fig 4.13: Stop Sign -1



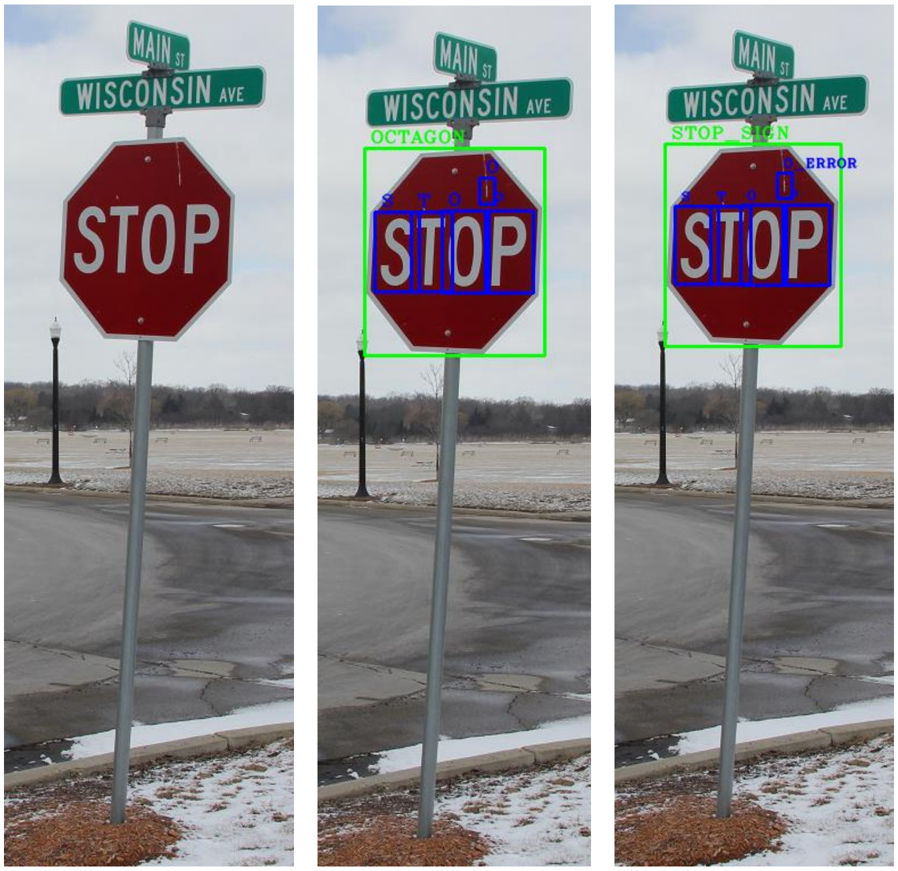
Original Contour-labels Object-model

Fig 4.14: Stop Sign – 2



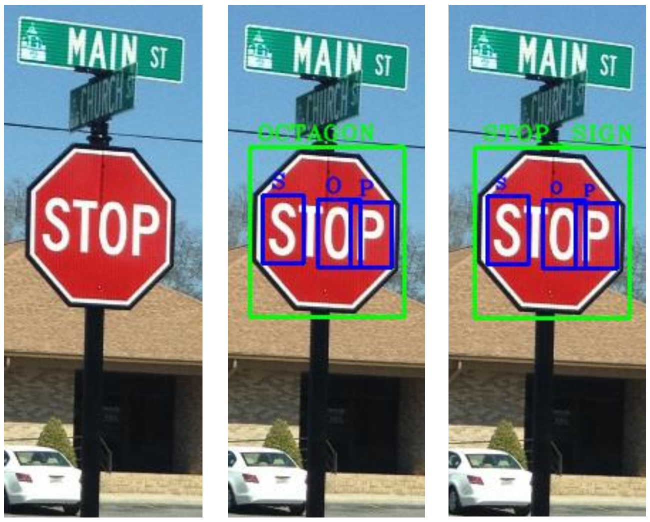
Original Contour-labels Object-model

Fig 4.15: Stop Sign - 3



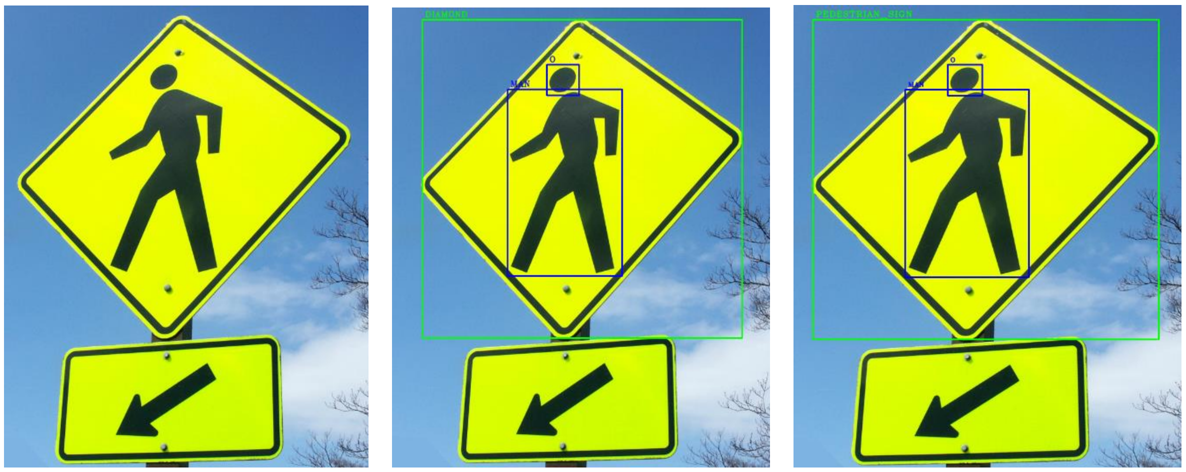
Original Contour-labels Object-model

Fig 4.16: Stop Sign – 4



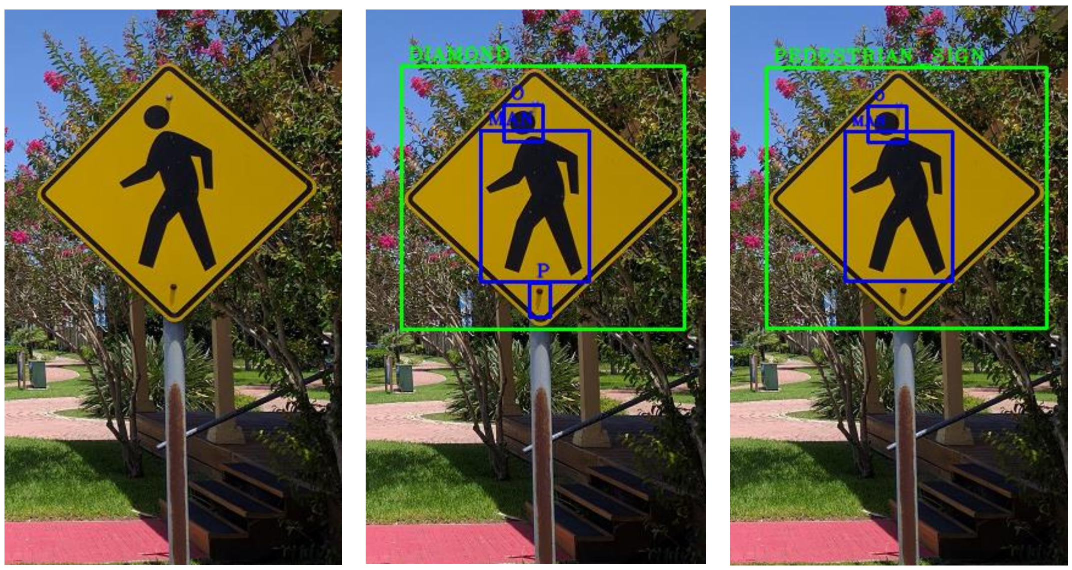
Original Contour-labels Object-model

Fig 4.17: Stop Sign – 5



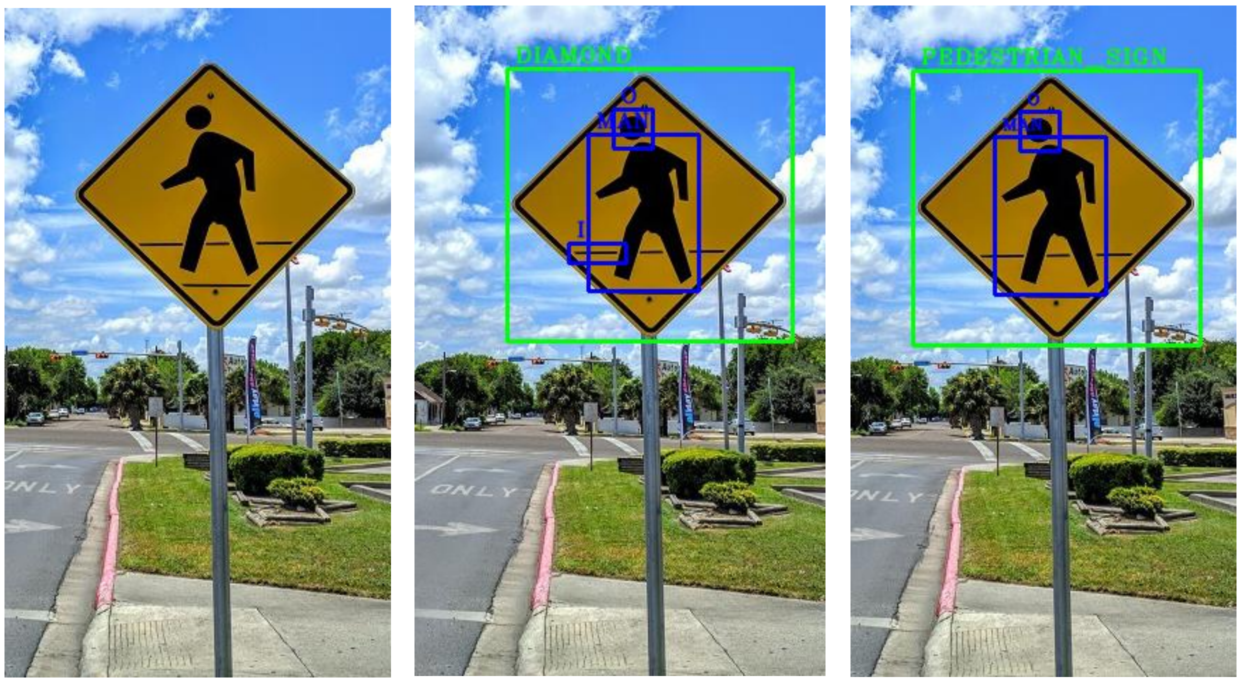
Original Contour-labels Object-model

Fig 4.18: Pedestrian Walk Sign – 1



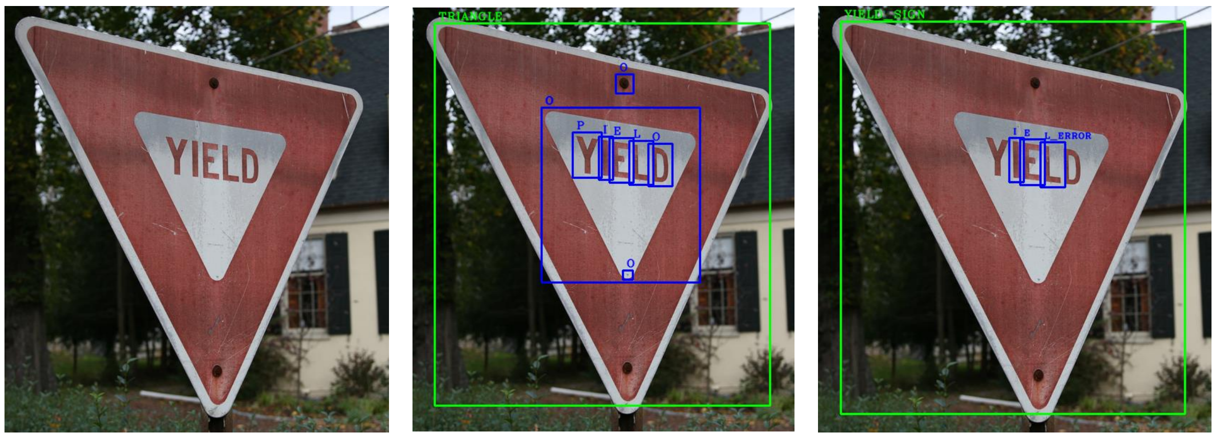
Original Contour-labels Object-model

Fig 4.19: Pedestrian Walk Sign - 2



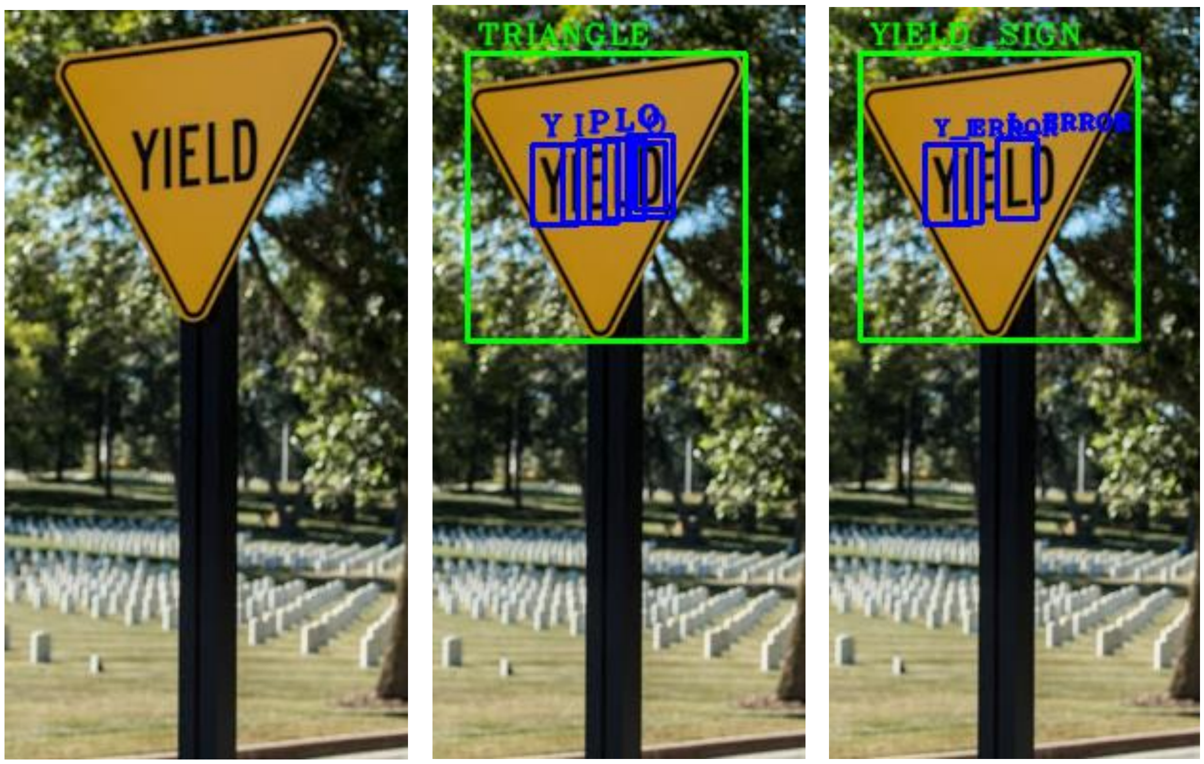
Original Contour-labels Object-model

Fig 4.20: Pedestrian Walk Sign - 3



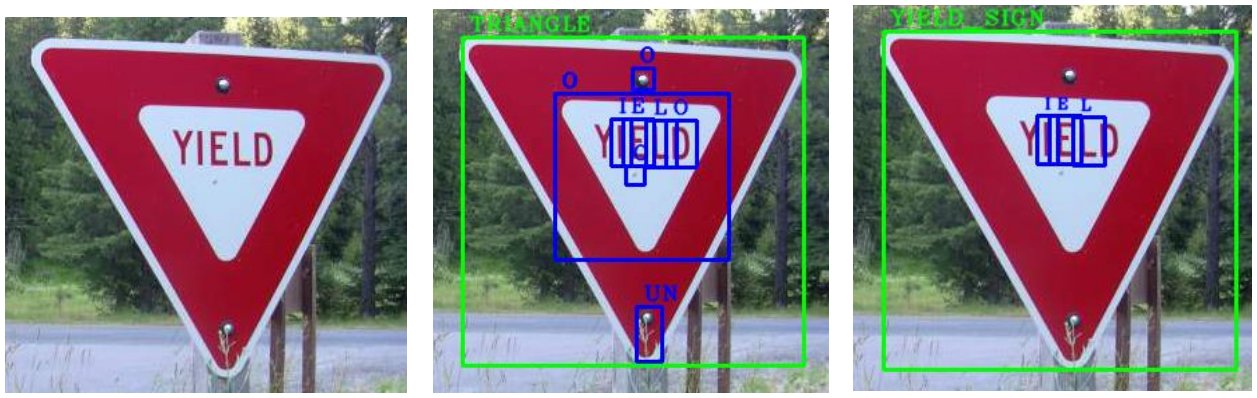
Original Contour-labels Object-model

Fig 4.21: Yield Sign - 1



Original Contour-labels Object-model

Fig 4.22: Yield Sign - 2



Original Contour-labels Object-model

Fig 4.23: Yield Sign - 3

CHAPTER V

CONCLUSION AND FUTURE WORK

The goals of this thesis were to implement a hierarchical method for contour extraction and designing an object recognition technique. The implemented technique further utilized a proven contour-description method, the Fourier descriptors, and popular classification algorithms were used to determine the shape labels. A simple hierarchical object-model matching algorithm was also implemented, along with the utilization of unique spatial characteristics of contours, to recognize objects. We built object-models of some common traffic signs and tested the technique on the same. From the experiments conducted, it was found that 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, with the FDs based method adding towards the robustness of the system.

Contours are features that represent vivid information about an object and its spatial characteristics. Contour detection is a difficult problem, that gets especially complex, when trying to consistently extract contours at different scales and in presence of textured backgrounds. Recently very good results have been achieved, by using deep features learned from convolutional neural networks, to detect contours [33]. More accurate contour detection methods, like the one referred above, will add to the consistency of the object recognition technique presented in this thesis.

An application to recognize traffic signs has already been presented in this work. 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 easily be adapted for real time object recognition and tracking purposes. A wide variety of applications can be thought of, especially where the contours of the object being recognized remain relatively consistent. The use of the OpenCV library makes it very easy to adapt to Real Time Operating Systems (RTOS) and other mobile operating platforms, like Android *etc*.

There is a lot of scope for both future enhancements and collaborations with this technique. At the feature description level, both local and global color and interest-point based features, can be used in conjunction with shape descriptors, to come up with a more detailed feature vector for the object. Learning algorithms like deep-neural networks can be trained and tested based on the feature descriptors derived from this technique. An immediate work item could be an efficient GPU implementation of the minimum distance measure used to classify the FDs.

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] Harris, C., & Stephens, M. (1988). A combined corner and edge detector. *In Proceedings of the 4th ALVEY vision conference,* (pp. 147– 151).

[3] 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.

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

[5] D. Ziou & S. Tabbone (1997). Edge detection techniques—an overview. *International Journal of Pattern Recognition and Image Analysis*, Vol. 8, (pp. 537 - 559).

[6] J.F. Canny (1983). A variational approach to edge detection. *Proceedings of the National Conference on Artificial Intelligence*, AAAI Press,Washington, D.C, 1983, pp. 54–58, (August).

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

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

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

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

[11] M. Hu (1962). Visual Pattern Recognition by Moment Invariants. *IRE Transactions on Information Theory*, IT-8:179–187, 1962.

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

[13] Freeman, H. (1961), On the Encoding of Arbitrary Geometric Configurations, *IRE Transactions on Information Theory*, EC-10(2), pp. 260–268, 1961.

[14] Freeman, H. (1974), Computer Processing of Line Drawing Images, *ACM Computing Surveys* ,6(1), pp. 57–95, 1974.

[15] S. Belongie, J. Malik, & J. Puzicha (2002). Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[16] Granlund, G.H. (1972). Fourier preprocessing for hand print character recognition. *IEEE Trans. Comput.*, 1972, C –21, (2), pp. 195– 201

[17] Zahn, C. & Roskies, R. Fourier descriptors for plane closed curves. *IEEE Trans. Comput*., 1972, C-21, (3), pp. 269–281

[18] 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.

[19] Mark S. Nixon & Alberto S. Aguado (2008). *Feature Extraction & Image Processing*, Second edition, Elsevier Ltd, 2008, pp. 285 - 311

[20] R. N. Bracewell (1986). *The Fourier Transform and its Applications*, McGraw Hill.

[21] 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.

[22] 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.

[23] Persoon, E., Fu, K.-S (1997). Shape discrimination using Fourier descriptors. *IEEE Trans. Syst.* Man Cybern., 1977, 7, (3), pp. 170 –179

[24] 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.

[25] J. Hopkins & T. L. Andersen(2005). A fourier-descriptor-based character recognition engine implemented under the gamera open-source document-processing framework. *DRR*, pages 111–118, 2005.

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

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

[28] Itseez (2015). Open Source Computer Vision Library - OpenCV.org, *https://github.com/itseez/opencv.*

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

[30] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, Mario Cifrek (2012). A brief introduction to OpenCV. *MIPRO*, 2012: 1725-1730.

[31] Deok J. Park, Kwon M. Nam and Rae-Hong Park (1995). Multiresolution edge detection techniques. *Pattern Recognition*, Vol. 28, No.2, pp.211-229.

[32] Takuya Minagawa (2014). DataAugmentation ver1.0. *https://github.com/takmin/DataAugmentation* .

[33] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai & Z. Zhang (2015). DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 3982-3991.

BIOGRAPHICAL SKETCH

Bassam Syed Arshad was born in Dammam, Saudi Arabia, to Indian parents, in 1987. He attended the International Indian School in Dammam, for his primary and middle school education. He moved, along with his family, to Dehradun, India, in early 2000, and attended the St. Thomas' College in Dehradun, for the remainder of his school education, graduating in 2004.

He completed his bachelors in electronics and communications engineering from the HKBK College of Engineering, Bangalore, affiliated to the Visvesvaraya Technological University (VTU), in 2011. He then worked for Dell International Services, Bangalore, for the next three years, as a Software Development Analyst. He left his job, at Dell, in July 2014, in pursuit of a master's degree in computer science, and started at the erstwhile UTB in the fall of 2014, which later merged with UTPA in fall 2015, to become the UTRGV. He graduated, with a MS in computer science, in August 2016, from the UTRGV. He can be reached at his permanent email address: bassamarshad@gmail.com.