# Understanding Learning Progressions via Automatic Scoring of Visual Models

**by**

**Ari Sagherian**

**A thesis submitted in partial fulfillment**
**of the requirements for the degree of**
**Master of Science**
**(Data Science)**
**in the University of Michigan-Dearborn**
**2020**

**Master's Thesis Committee:**

**Assistant Professor Mohamed Abouelenien, Chair**
**Professor Bruce Maxim**
**Professor Qiang Zhu**
**Chee Wee Leong, Educational Testing Service**

**Acknowledgements**

I would like to thank Suhasini Kalaiah Lingaiah and Blake LaFuente for their help in planning and implementing this project.

## Table of Contents


Acknowledgements. . . . . ii

List of Tables. . . . . v

List of Figures. . . . . vi

Abstract. . . . . viii

Chapter 1 Introduction . . . . . 1

Chapter 2 Related work . . . . . 4

2.1 Conceptual Background . . . . . 4

2.2 Color Segmentation. . . . . 5

2.3 Contour and Edge Detection . . . . . 9

2.4 Template Matching . . . . . 11

2.5 Hu Moments . . . . . 13

2.6 Hough Transform . . . . . 15

2.7 Gaussian Blur . . . . . 16

2.8 Machine Learning Approaches . . . . . 17

2.9 Deep Learning Approaches. . . . . 20

2.10 Modern Approaches to Shape Classification . . . . . 21

Chapter 3 Dataset . . . . . 24

Chapter 4 Methodology. . . . . 27

4.1 Template Extraction . . . . . 28

4.2 Gaussian Blur. . . . . 28

4.3 Color Segmentation. . . . . 29

4.4 Contour Detection and Shape Approximation. . . . . 30

4.5 Rules-based Shape Classification . . . . . 32

4.6 Hu Moment Classification . . . . . 33

4.7 Area Ratio Classification. . . . . 35

4.8 Template Matching. . . . . 35
4.9 Machine Learning Classification . . . . . 37
4.10 Cascaded Voting System. . . . . 38
4.11 Arrow Orientation . . . . . 41
4.12 Scoring of Visual Models . . . . . 46
Chapter 5 Experimental Results . . . . . 49
5.1 Object Classification Results. . . . . 49
5.2 Learning Progression Scoring . . . . . 54
5.3 Cross Domain Analysis . . . . . 59
Chapter 6 Discussion. . . . . 62
Chapter 7 Conclusion . . . . . 65
References . . . . . 66

## List of Tables

4.1 Mapping Learning Progressions to four dimensional sub-progressions . . . . . .46

4.2 Descriptions of 10 features used to score visual models . . . . . 47

5.1 Precision, Recall, and F1 Score on the 50 Ocean Water dataset . . . . . 53

5.2 Precision, Recall, and F1 Score on the 144 Ocean Water dataset . . . . . 54

## List of Figures


2.1 Canny edge detection on raw image (A) and Gaussian blurred image (B). . . . . .10

2.2 Hough Line Transform at thresholds (A) = 35 and (B) = 75. . . . . .16

2.3 Gaussian blur with increasing window sizes. . . . . .17

3.1 Visual models from the Ocean Water dataset . . . . . .25

3.2 Visual models from the Synthesized Ocean Water dataset . . . . . .26

3.3 Visual models from the Two Can dataset. . . . . 26

4.1 Overview of methodology . . . . . 27

4.2 Examples of template images . . . . . .28

4.3 3 by 3 Gaussian blur matrix. . . . . .29

4.4 Contour and bounding shape examples. . . . . .32

4.5 Template matching for rocks with bounding box shown. . . . . .37

4.6 Flow chart of the cascaded soting system . . . . . .40

4.7 Adjusted arrow orientation based on 3-degree margin of error. . . . . .43

4.8 Adjusted arrow orientation based on coordinate similarity. . . . . .44

4.9 Adjusted Northwest arrow orientation . . . . . .45

5.1 Accuracies of classification methods on the 50 image Ocean Water dataset . . . . . .51

5.2 Accuracies of classification methods on the 144 image Ocean Water dataset . . . . . .51

5.3 Low accuracy visual model due to non-white background. . . . . .52

5.4 QWK Scores for 8 learners on the 144 image Ocean Water dataset . . . . . 55

5.5 144 Image Ocean Water learning curves for voting system (A) and rules-based (B) methods. . . . . 56

5.6 QWK Scores for 8 learners on the 174 image Ocean Water dataset . . . . . 56

5.7 174 Ocean Water learning curves for voting system (A) and rules-based (B) . . . . . .57

5.8 QWK Scores for 8 learners on the Two Can dataset . . . . . 58

5.9 Two Can learning curves for the voting system (A) and rules-based (B). . . . . .58

5.10 Cross domain QWK trained on Ocean Water dataset and tested on Two Can dataset . . . 60

5.11 Cross domain QWK trained on Two Can dataset and tested on Ocean Water dataset . . . .61

## Abstract

The modern reliance on technological advances has spurred a focus on improving scientific education. Fueled by this interest, novel methods of testing students' understanding of scientific concepts have been developed. One of these is visual modeling, an assessment method which allows for non-textual evaluation that incorporates previously difficult factors to test, such as complexity and creativity. Although visual models have been shown to effectively measure conceptual understanding, there has been a logistical barrier of scaling due to the infeasibility of grading large amounts of them by hand. This thesis proposes a system that can solve this issue by automatically grading visual models. A host of unsupervised and supervised computer vision techniques are utilized in order to classify shapes in visual models, extract relevant features, and, ultimately, assign a Learning Progression score to each model. Examples of the techniques used are a novel way to determine the orientation of Arrows and a Cascaded Voting System for shape classification. The results of the automatic grading system proposed in this thesis outperform previous methods and lay the foundation for future improvements. The resulting findings show great promise for directly solving the scaling issue, thereby making visual model assessments a practical tool for widespread use.

## Chapter 1: Introduction

The modern world is progressively growing more reliant on advances in technology, creating an international race for innovation. As explained by Augustine et al. [4], America has been falling behind other countries in key industries involved in science, technology, and medicine. One of the core suggestions to remedy this situation has been to emphasize education in fields related to innovation, making the notion of a Science, Technology, Engineering, and Mathematics (STEM) education more popular. Along with this reinforced emphasis on STEM subjects came the issue of assessing students' progressive growth and understanding of these concepts. Focusing strictly on the Science portion, assessment experts aimed to find measures that could evaluate the multiple dimensions of scientific knowledge such as theoretical concepts and experimental practices [53]. Learning progressions (LPs) were developed to facilitate the continual assessment of students' progress in scientific knowledge acquisition [16]. These LP's were aggregate scores of sub-progressions, in which each sub-progression focused on a specific dimension of the overarching concept. For example, theoretical ideas and practices could each be a sub-progression of a general concept and the amalgam of both sub-progressions would reflect the overall LP.

In accordance with the push for better STEM education, the Next Generation Science Standards (NGSS) specifically designated visual modeling to be a valuable skill for students between grades K-12 to have [54]. These visual models help capture the totality of students' understanding of scientific knowledge, as opposed to the narrower verbal-based tests commonly

used. To address the challenge of assessing visual models, LP's were created focusing on visual models illustrating the state and behavior of Matter [51, 25]. The initial development of LP's for assessing students' understanding held potential but had a core flaw. The visual models were graded by hand, quickly running into the economic infeasibility of scaling such a system to larger pools of students.

The advent of Computer Vision (CV) and Artificial Intelligence (AI) tools in recent years has unlocked the potential to automatically grade these visual models, essentially solving the scaling problem. The first attempts at this were conducted by Leong et al. [38] in their creation of an automatic grading system of visual models. The assessment was mapped to the previously developed LP, revealing a promising way forward. Although this approach seemed to be a solution to the scaling problem, it still had a shortcoming. The visual models were drawn in a JavaScript Object Notation (JSON) file format that automatically labeled features about the objects as they were created. The assessment was then based on these self-labeled, extracted features. Although this method worked for the specific study, there remained the inability to derive these features from general images outside of the JSON format.

This study has two primary goals: (1) to accurately classify objects and extract their features from visual models stored as unlabeled images and (2) to use these classifications and features to automatically assess students' understanding of a concept based on the concept's LP.

The overarching motivation behind these goals is to be able to generalize the previous research on automatically grading visual models and overcome the inherent scaling issue. We accomplish this by first using a multitude of unsupervised CV techniques to identify the objects and features within each model using different datasets provided by Educational Testing Service (ETS), namely the Ocean Water and Two Can datasets. Moreover, we implement a supervised

Machine Learning approach in order to compare with the unsupervised approaches. Then, a cascaded voting system is developed to integrate the classifications of individual methods in order to improve the final decisions made by the framework. The resulting classifications and extracted features are then aggregated to 10 higher level features. Finally, these aggregated features are used in the grading system developed by Leong et al. [38] to predict the LP score of each visual model, ultimately resulting in a fully autonomous and scalable approach to assessing students' conceptual understanding of scientific knowledge.

This paper is organized as follows. Chapter 2 reviews the related work to this research, including the conceptual background to this topic and the technical tools used in similar tasks. Chapter 3 describes the datasets comprised of visual models used in this project. Chapter 4 explains the methods used for each step of the project. Chapter 5 discusses the experimental design and results, comparing the classification accuracies and LP predictions of the many approaches. Chapter 6 discusses the findings of this project and highlights paths for future developments. Finally, Chapter 7 summarizes and concludes the findings of this project. Throughout this paper the terms "visual models" and "images" are used interchangeably.

## Chapter 2: Related Work

The conceptual foundation and a wide variety of computer vision techniques are explored in this section. The organization of this section will be of subsections for different unsupervised techniques, Object Recognition, Edge Detection, Color Segmentation, and Template Matching, as well as supervised Machine Learning and Deep Learning approaches, among others. The subcategorization is used for simplicity in organization only, as many of these techniques are frequently used in combination to achieve a specified goal.

### Section 2.1 Conceptual Background

The conceptual foundation of this project was created by Leong et al in their research on grading scientific visual models. They administered questions to students who, in response, created visual models to illustrate their understanding. These models were then evaluated by a four-dimensional Learning Progression (LP) with levels ranging from 1 (lowest) to 5 (highest) [16]. The data used in this study was the set of visual models created by the students in their respective assessment tasks. Each model consisted of multiple 2-Dimensional shapes arranged to illustrate a concept. Features of each shape, such as the type of shape, the color, the coordinates, and the rotation were extracted and split into counting-based and spatial-based features. Each of the four dimensions in the LP had their own sub-progression of 5 levels, with the amalgam of the four dimensions resulting in the final LP score. The Scale, Behavior, and Material Identity dimensions were based on the counting-based features and calculated based on their respective deviations from

expected counts. The fourth dimension, Density, was determined by applying the *k*-nearest Neighbors algorithm to the spatial-based features.

Evaluation of the 148 visual models was conducted with a 10-fold cross validation. For each model, 10 features were extracted based on the counting-based and spatial-based features. Then, commonly used regressors and classifiers were used to predict sub-progression levels for the four dimensions. After the levels of each of the four dimensions were determined, the final LP value was calculated based on specific combinations of these scores. Although this study provided the conceptual groundwork, much room for improvement remained. The primary flaw of this study was that the evaluation methods were completed by people. As the number of people this assessment is administered increases, the ability to manually grade all the models grows more infeasible in addition to the limitation discussed earlier using the JSON files. To address this core issue, this project will aim to automate the grading to allow for broader use of visual model assessments. In order to accomplish this, unsupervised Computer Vision methods will be compared with supervised Artificial Intelligence methods to experimentally determine the optimal approach.

### Section 2.2 Color Segmentation

Images can be represented in a variety of color spaces where a color space is a tuple of integer values representing the resulting color of a pixel. In this project, images are represented in the Red, Green, and Blue (RGB) color space. Each of the component colors has discrete values ranging from 0 to 255 denoting the respective color intensity. These individual values combine into a 3-value tuple in which (0, 0, 0) represents the color black, as it is devoid of all color, and

(255, 255, 255) represents the color white. The multiple combinations of all the varying values of each component thus produces the visible light spectrum in accordance with the photoreceptors of the human eye.

Image segmentation is the process of separating an image based on specific categories such as color, region, and many other criteria. Automatic image segmentation techniques have been widely developed since the onset of Computer Vision and can be broadly categorized into four sections: 1) thresholding techniques, 2) boundary-based methods, 3) region-based methods, and 4) hybrid techniques [22].

Threshold segmentation is based on the idea that pixels in an image within a certain range of color values are part of the same class [57, 39]. Using the RGB color space as example threshold segmentation can be implemented by deciding an arbitrary difference between color values. Then, pixel values will be compared with each other and pixels within the difference value will be grouped in the same class. Thresholding works well when there are objects that have stark differences in color. Issues begin to arise when the boundaries between objects are blurry and multiple objects have slight variations in color. In summary, thresholding segmentation remains a powerful tool when objects have stark differences in color.

Boundary based methods are similar to thresholding but take advantage of the stark changes in pixel values at the boundary between two objects [58]. Searching for drastic changes has the benefit of isolating individual shapes rather than separating based on arbitrary color differences like thresholding. This becomes highly effective for detecting the presence and outline of objects. For this reason, many edge detection algorithms such as Sobel and Canny use boundary-based methods which will be explored in Section 2.3. A shortcoming of these approaches is the susceptibility to overlapping images. For example, if a circle is overlapping a square, it may detect

where the boundary is, but it will leave the square with an incorrect bounding region which can have downstream consequences when the shapes are attempted to be classified.

Region based methods use the opposite assumption of boundary-based methods. Instead of assuming that stark changes signify different classes, region based methods assume that adjacent pixels of the similar values are in the same class. A region based approach that works particularly well for homogenous images is the split and merge algorithm. This works by recursively splitting quadrants of an image, testing for a user-defined homogeneity criterion, then merging similar portions with their neighbors until the whole image passes the homogeneity test [31]. The issue with this is when non-homogenous images are present, which is common for complex shapes and real images.

Finally, Hybrid methods are combinations of boundary based and region-based methods. They take the advantages of both detecting boundaries to distinguish classes and recognizing similarities in pixel values to classify a specific region. The dual approach allows for a better segmentation of an image based on a desired criterion, such as color for our project.

An example of an unsupervised image segmentation method is the JSEG algorithm which is split into two major steps: color quantization and spatial segmentation [19]. The benefit of decoupling color and spatial segmentation is that a complex problem can be broken down to two simpler ones, each with more specialized algorithms. The first step is to quantize the colors into an average of 15 class-labels based solely on their color space values, disregarding their spatial location. Then, the pixel values are represented by their class labels, forming a class-map. Finally, this class-map can be regionally segmented without directly considering the similarity between the pixel values. The segmentation is completed by assigning a J-value to each class-map, which is a measure of the homogeneity of the class-map values in the image. The J-value is proportional to

the homogeneity of an image, meaning that the more clustered the class-maps are, the higher the J-value will be. Conversely, the more evenly distributed the class maps are, the lower the J-value will be. The goal for region segmentation then becomes to minimize the J-value for each region to increase homogeneity.

An additional consequence of applying this method is that non-homogenous portions of images near boundaries will have higher J-values. Region determination is conducted by picking seed locations based on local J-value minima, and then recursively expanding the region until large J-values are reached. The hybrid nature of the JSEG algorithm emerges at this point, with a region-based approach for J-value minima and a boundary-based approach for high J-values. Experimental results were conducted on 2,500 Corsel stock photos and evaluated against human labelled color segmented images. The results were demonstrated to be subjectively good based on the boundaries shown in the images, however, the primary shortcoming of this study is the lack of ground truth labels for objective measures. Furthermore, other studies have pointed out that the results suffered from over segmentation on occasion. This has the potential consequence of missing the connectedness of a segment, essentially missing the forest and focusing on the trees [33].

Returning to threshold segmentation, an initial issue was the lack of adaptivity for various images. Choosing an arbitrary threshold level to differentiate classes of objects may work well for a specific image but can have disastrous results on another [71]. To address this problem, Littman et al. [42] conducted a review on the various adaptive thresholding techniques, particularly comparing neural networks and statistical approaches. Their goal was to test which methods were superior by testing them on a binary classification task. They analyzed 80 images of the same hand from different perspectives and in eight different background scenes. The task at hand was to

classify each pixel as either belonging to the “hand” or the “background” class. They demonstrate that both statistical methods and neural networks are effective at this task, but neural networks are slightly better. Although this study only focused on a binary classification, the general overview can help guide decisions of what type of color segmentation techniques to implement.

## Section 2.3 Contour and Edge Detection

The discernment of distinct shapes in an image can be accomplished by detecting the edges or contours of each shape. A contour is a curve of continuous points that are one pixel wide, forming a boundary if the contour is closed. These boundaries can then be analyzed to classify them into specific shapes.

Contour detecting algorithms are generally judged on four metrics: 1) accuracy of contour tracing, 2) computing time, 3) size of data, and 4) restoration of contour based on saved data. Many approaches have been developed based on the foundational work by Suzuki et al [74] which laid out the conceptual framework for contour detection, but most either failed or incompletely addressed the four criteria [14, 15]. An approach that attempts to address all four of these criteria was proposed by Seo et al [67]. They leveraged the same core ideas as Suzuki et al and improved it mostly at the corner detection.

In order to compare their results with prior contour detectors, they tested their approach on the nine Consultative Committee for International Telephony and Telegraphy fax images. Their results showed approximately 99.5% accuracy along with the highest pixel count for contours detected, illustrating that their approach surpassed prior versions at detecting the corners of contours. Furthermore, the process was computationally fast and had built-in data compression

which can be useful depending on the context of the project. This approach holds much promise for this project due to its high accuracy and ability to better detect corners. However, the small sample size of nine images induces skepticism into the robustness of the approach when applying to other types of images.

Other research methods have been used historically include edge detection which holds promise for their application to object recognition [2, 3, 6, 18, 40, 44, 46, 55, 59]. Three of the original edge detectors were the Sobel [72], Roberts [64], and Prewitt [61] operators. These all aimed to detect edges by using local derivative kernels and applying these across the entirety of a grayscale image.

Another famous edge detector is the Canny Edge Detector [13]. The Canny detector finds discontinuities in the intensity of pixels then adds non-maximum suppression and hysteresis to detect the maximum number of edges without repeating them [48]. Non-maximum suppression is an edge thinning technique which helps determine a single edge with the greatest change in gradient intensity. Noise still exists in the image at this point so a double threshold (high and low) is applied, classifying potential edges with higher values than the high threshold as “strong edges” and those below the low threshold as “weak edges”. Finally, hysteresis is applied to filter out “weak edges” that are not actual edges via a simple filtering process. The hysteresis mechanism operates in that if a “weak edge” pixel is neighbors with a “strong edge” pixel, then it will be maintained. If not, then it will be removed to reduce the noise in the edge detector [45, 60, 13].

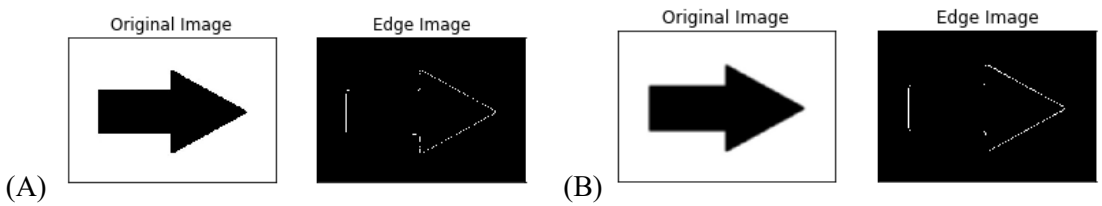


Figure 2.1: Canny edge detection on raw image (A) and Gaussian blurred image (B)

The practical use of Contour and Edge Detectors in the context of shape classification is typically a pre-processing step. Contour detection produces closed loops of continuous pixels and edge detectors produce the clearest edges of a shape. These can both be used, depending on the context, to produce candidate shapes which will undergo further analysis in order to ultimately classify them.

## Section 2.4 Template Matching

Template matching is the process of detecting if and where a template pattern is found within another image. In general, template matching approaches can suffer from three issues. The first is that the template might be scale variant, meaning that if the desired object is found either larger or smaller than the scale designated in the template, it will not register as a match. The second flaw is that some approaches are rotation variant. This means that even if the objects are the same size, a difference in orientation can result in a false negative. These two flaws alone render simple template matching useless in nearly any context with variations in shapes/objects. The third issue is the great computational cost to sequentially calculate the difference in a template pattern and an image.

Many algorithms have been developed to address the large processing time of the Full Search approach which calculates the differences in the template pattern sequentially. A variety of assumptions are the starting point for many of these. For example, instead of using the entire template image, approximating the pattern based on key features [75] or reducing the search space [30] can reduce the amount of computation. Another effective method called Partial Distortion Elimination utilizes a threshold, with the assumption that if the dissimilarity value between a

template and the target exceeds the threshold, then the comparison is halted, and the image is deemed a mismatch [8]. These approaches and many more outlined in the review by Ouyang et al. all uniquely solve the processing speed problem, yet they fail to solve the scale and rotation shortcomings [56].

A solution to the scale and rotation problems would be to begin with a large template image and progressively scale it down while checking for each rotation angle at each scale. This cumbersome, brute force method holds the possibility for high accuracy at the cost of great computation time. In order to minimize the computational cost, the Ciratefi Algorithm was created by Kim et al. which aims to have the same accuracy as the brute force method at a fraction of the time [35]. The Ciratefi Algorithm uses three cascaded filters that successively filter out pixels that bear no chance of matching the template. The first filter, called Cifi, determines the probable scale factor of the detected objects. Then, the second filter, called Rafi, assigns a probable rotation to the object. The third and final filter, called Tefi, applies standard template matching at the scale and rotation factors from the Cifi and Rafi filter.

To test whether the Ciratefi algorithm truly matched the brute force method at a fraction of the time, 145 images were analyzed with shapes of varying rotations, scales, brightness, and contrasts. They conducted 3 experiments: detecting 2-D shapes, detecting McDonald's signs, and detecting buildings from Google Earth. The shape detection experiment, which is the most relevant to this project, produced a perfect accuracy of 700 out of 700 correct detections. The detection of the McDonald's sign had 114 out of 116, with the errors likely due to high levels of brightness and not incorporating color information. The building detection had 171 out of 187 correct, with errors likely due to shadows, occlusions, and high variance in brightness. With such high accuracy,

especially for shape detection, the Ciratefi algorithm holds great promise and can possibly be improved upon by incorporating color information and devising methods to account for occlusions.

## Section 2.5 Hu Moments

Moments in Computer Vision are calculations that describe features of a binarized image or contour. Focusing on the context of contour-based shape recognition, moments can offer information about the pixel intensity, location, scale, rotation, and other features which can further be used to classify shapes. The various features extracted by moments are developed hierarchically, with newer methods incorporating prior versions. The first level of moments, called $0^{th}$ order moments, simply calculate the sum of pixel intensity of a contour. In a binary image where black pixels have a value of 0 and white pixels equal 1, the $0^{th}$ order moment is simply the sum of all white pixels. This can describe the area of a contour but offers little more information because any change in the scale of the contour leads to completely different values.

Building on the $0^{th}$ order moment, raw moments calculate the sum of pixel intensities with reference to their location in an image. Raw moments offer little benefit because if a shape is translated at all, the returned value changes drastically. A big leap forward were Central Moments which achieved translation invariance by deriving the centroid, which is the weighted average of the pixel intensities, and subtracting this centroid from the Raw Moments of the image. The subtraction of the centroid allows similar values to emerge regardless of the location of a contour. Continuing to build on the developments, Normalized Central Moments were derived to achieve both translation and scale invariance. The addition of scale invariance was achieved by dividing the Central Moment by the area of the contour, which was calculated from the $0^{th}$ order moment.

As the information that moments returned grew, so did their use in object recognition. Although Normalized Central Moments offered scale and translation invariance, discrepancies between matching shapes emerged when one was rotated or reflected.

The groundbreaking creation of Hu moments remedied these last two flaws, achieving the valuable properties of translation, rotation, scale, and reflection invariance. These properties make Hu moments of particular interest for this project as multiple images are scaled, translated, and rotated multiple times per image. Hu moments are a set of seven values that expand upon Normalized Central Moments while preserving the geometric properties of an image under rotation and skew [32, 24]. The results of these Hu Moments can then be extracted as feature vectors for further analysis to determine the similarity between shapes.

Multiple examples exist for shape recognition based on the features determined by Hu and other types of moments [77, 76, 23]. The general schema for these object recognition methods is to first extract features of the image using moments. These features are then used as inputs into a classifier algorithm that determines the class of the shape. Some of the common classifiers used are *k*-Nearest Neighbors, Multilayer Perceptrons, various Neural Networks, and fuzzy *k*-Near Neighbors. Reviewing their performances, high accuracy was achieved for the Multilayer Perceptron, Neural Network, and the *k*-Nearest Neighbor classifiers which will further be explored in Section 2.8 [50].

Many of these studies begin by pre-processing the images to either binary or grayscale. An example of this is the research conducted by L.A. Torres et al., which first preprocessed the images by binarizing them to black and white. Then, the authors extracted features based on moments and used the extracted features in a holographic Nearest Neighbors (HNN) algorithm. An HNN is similar to the K-Nearest Neighbors algorithm but slightly faster and more accurate. They applied

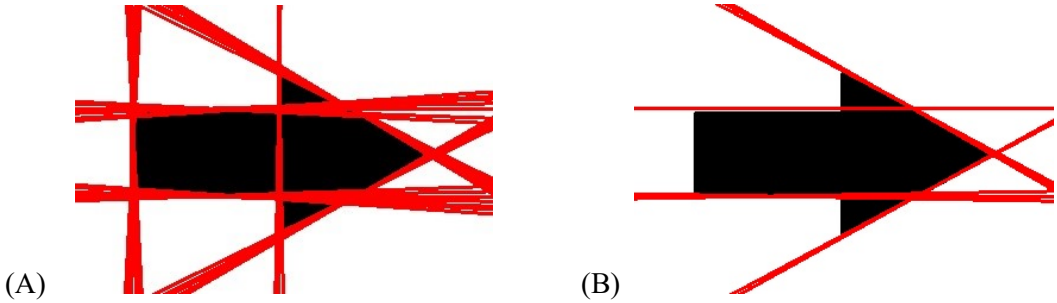
their architecture to identify the 26 letters of the English alphabet with 107 training instances and 6,188 testing instances. The testing instances were a combination of 17 different scales and 14 rotations for each letter, i.e. 238 instances for each letter. The results were between 95-100% accurate depending on the letter and the amount of transformation applied to it. This holds promise for our project because it shows that the utilization of Hu moments in combination with various classification algorithms can yield high accuracy in recognizing shapes.

## Section 2.6 Hough Transform

Hough Transform is a robust computer vision technique used to detect lines and curves within an image. The primary advantage of the Hough Transform is that it is occlusion insensitive, an issue that has lowered the accuracy of object detection for certain shapes. An additional benefit of the Hough Transform is that it detects lines in polar coordinates, giving valuable rotation information. The first step in applying the Hough transform is to implement an edge detection algorithm such as Canny to produce the coordinates of the boundaries. Once the edges are determined, the Hough Transform connects these edges into lines based on a threshold. The lower the threshold, the higher the probability that the edges connect as a line, resulting in an inverse relationship between the threshold and the number of lines created [21]. A result of this relationship is illustrated in Figure 2.2 from one of the template shapes extracted from our datasets.

Hough Transform is not restricted to simply detecting lines in images though. By generalizing its properties and combining with other classifiers, the Hough Transform can extract features that are then used for shape detection similar to the use of Hu moments [5]. Although Hough Transform works well for detecting shapes, it performs comparatively worse than using Hu

moments when it comes to classifying them [70]. Nevertheless, the Hough Transform can be a powerful tool for determining the orientation of shapes since each line is calculated in polar coordinates, a feature of potential great benefit.



(A) (B)

Figure 2.2: Hough Line Transform at thresholds (A) = 35 and (B) = 75

## Section 2.7 Gaussian Blur

A common pre-processing step is to filter an image using a Gaussian blur. The Gaussian blur is a simple method with the ultimate goal of improving the signal to noise ratio, a key step that shows benefits in down-stream applications. The process of applying a Gaussian blur is to create a 2-D kernel which convolves an image while applying a Gaussian function as it proceeds [68]. The larger the kernel, the more blurry the image will appear as shown in Figure 2.3.

The result of this will be that each image will acquire a value equal to the weighted average of the neighboring pixels, with further pixels having smaller weights and the original pixel having the heaviest weight. After processing the entire image, the noise from artifacts and other mistakes is reduced, allowing for better applications such as color segmentation, edge detection, and object recognition.

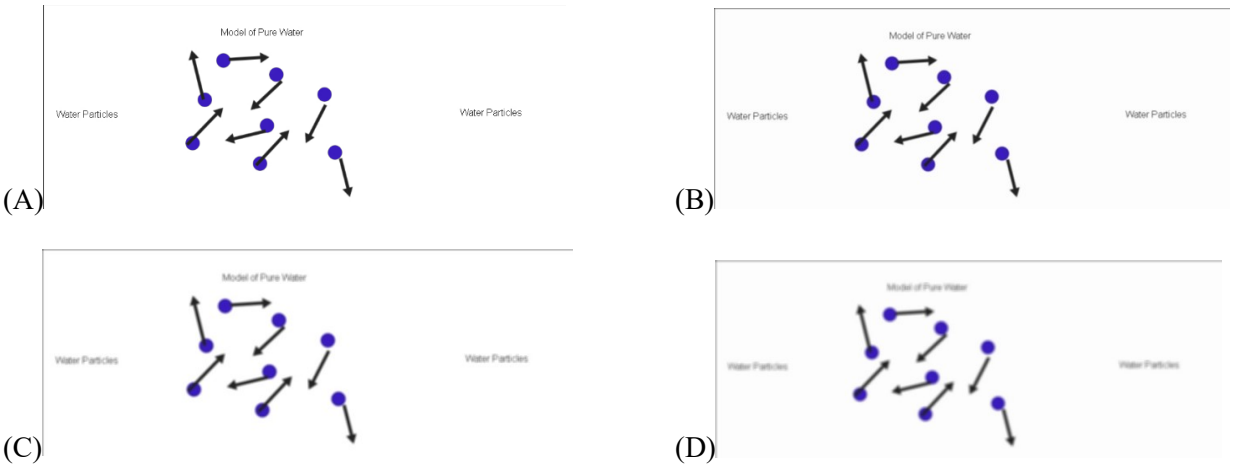



Figure 2.3: Gaussian blur with increasing window sizes
(A) original image, (B) 3x3 window, (C) 5x5 window, and (D) 7x7 window

One parameter that can be tweaked in the Gaussian function is the standard deviation. A study conducted by Gedraite et al. showed that, in general, larger standard deviations in the Gaussian function tend to produce better results for image segmentation, which is highly relevant to this project [26]. Furthermore, they showed that small standard deviations are useful for images with increased noise. This research lays out the conceptual trend of how to adjust the parameters of a Gaussian Blur, giving a heuristic for use instead of blind trial and error.

## Section 2.8 Machine Learning Approaches

Machine learning algorithms have been applied to many branches of computer science and computer vision. The main branch of machine learning algorithms applicable to this project are supervised classifiers. Supervised classifiers, in the context of object recognition, can generally be described as algorithms which receive features describing the object as an input. These feature vectors will be used to predict a certain outcome, for example classifying a certain shape.

Supervised approaches have labeled ground truths for a certain number of instances, allowing the algorithm to optimize itself to better predict the correct classification based on the present features. Once the program has trained itself on the labelled data, it can then be tested on novel instances to assess its performance.

The accuracy of using a machine learning approach has two main components that will affect the final outcome. First, the method of extracting quality features is critical. Second, the various machine learning classifiers used on these extracted features will determine the ultimate outcome. The three most prevalent feature extraction algorithms for object recognition are the Scale Invariant Feature Transform (SIFT), Speeded up Robust Feature (SURF), and the Oriented FAST and Rotated BRIEF (ORB) algorithms [43, 65, 7]. The reason for the prevalence of these three is that they extract features that are invariant to scale, rotation, translation, blur, and illumination [52].

The SIFT algorithm was the first of these and set the bar for effectiveness but lacked speed, which SURF and ORB both aimed to address. To test these three, Karami et al. [34] compared all three on a curated dataset of images that were distorted in various fashions such as by scale, rotation, blur, and intensity. The results show that the SURF and ORB algorithms are significantly faster than SIFT in every category, however, the accuracy is generally better with SIFT when used to match shapes. In particular, SIFT is superior for images with rotation and illumination changes.

The ORB algorithm, which was significantly superior for scaling differences, was the fastest of the three algorithms, and was slightly better than SIFT for blurred images. The SURF algorithm typically performed in between SIFT and ORB but was not the best in a single category [REFERENE Karami E.]. Based on these results, SIFT and ORB show potential for feature

extraction. For overall accuracy the SIFT approach is superior, however , if speed is of necessity, then the ORB approach is the ideal candidate.

After extracting features, different machine learning algorithms can be applied to match shapes. Two of the most commonly used are the *k*-Nearest Neighbors (KNN) and Support Vector Machine (SVM) algorithms due to their robustness [11, 17, 12]. To compare these two algorithms, four classes of images were pre-processed using the SIFT algorithm to form feature vectors and then the individual algorithms were applied to these feature vectors. A 5-fold validation test in which each fold had 2800 training and 700 test images was conducted. Multiple values were tested for the number of nearest neighbors for the KNN with the best being k-value equaling 5. At this k-value, the SVM outperformed the KNN algorithm with approximately 92% accuracy compared to 78% [36].

This is partially due to the KNN suffering greatly at classifying one of the classes of images. When this class was removed, the SVM still outperformed KNN but by a marginal 4 percent. A possible explanation for this is that the KNN approach gives equal weights to all features which can generate large discrepancies depending on the quality of the features. The large difference in performance when ignoring one of the classes showcases a flaw with the KNN approach, however, since it performs well on the other classes, KNN still may be useful. Testing the various combinations of these classifiers with the SIFT and ORB feature extractors have a compelling case for application purposes using modern machine learning.

## Section 2.9 Deep Learning Approaches

Recent advances in computational power and progress in Artificial Intelligence have given rise to the many applications of deep learning algorithms. For computer vision tasks in particular, a type of deep learning algorithm called a Convolutional Neural Network (CNN) has been widely used for a variety of applications. The prevalence of CNN's in computer vision tasks is rooted in the original idea of a CNN being modeled after the human visual system, in that there are multiple layers to the network with non-fully connected intermediate layers extracting information and finally feeding into a fully connected layer at the end. A key feature of CNN's is that they are feed-forward neural networks, meaning that each input is independently processed, with one input not impacting another input. Another feature is that CNN's are bound to producing an output layer with the same size as the original input layer.

The input/output size constraint can become a problem for object recognition because the number of objects to be detected is unknown prior to analysis. This causes variability in the output size depending on the input image. To remedy this issue, Girshick *et al.* created the Region-based Convolutional Neural Network (R-CNN) algorithm. The R-CNN algorithm first uses a Selective Search algorithm to break the image into 2000 region proposals and then implements a CNN on each of those regions to detect objects [28] rich feature]. Applying the CNN to each region individually remedies the issue of variable inputs since each region will have one object. To test their algorithm, the PASCAL VOC collection of datasets was used, and the results were significantly improved compared to prior approaches, reaching mean average precision of 62.4% on the VOC 2012 dataset.

The R-CNN had two main flaws with it. First, the algorithm was very slow, making it impractical for real-time usage. The second, is that the Selective Search algorithm used for region

detection was a fixed approach, having no machine learning or adaptation. To improve on these, the Fast R-CNN was developed by Girshick *et al.* The core difference between the two approaches is that the Fast R-CNN does not have to generate all 2,000 region proposals each time. Instead, the input image is fed to a CNN which then returns a feature map from which the Selective Search algorithm creates region proposals. This leads to potentially fewer regions being proposed and saves computational time. Finally, objects are then detected in similar fashion to the original R-CNN. The Fast R-CNN increased the training and performance speed many fold compared to the R-CNN and even raised the mean average precision on the VOC 2012 dataset to 68.4% [27].

However, room remained for improvements in speed and accuracy, with the region proposal step being the main bottleneck for processing time. The next evolution of this approach was the Faster R-CNN. The Faster R-CNN approach replaced the Selective Search algorithm with a Region Proposal Network (RPN). The RPN was then combined with the Fast R-CNN by sharing convolutional features, essentially guiding the Fast R-CNN detector to the regions of interest. This led to increases in speed and higher mean average precision, with a score of 75.9% on the VOC 2012 dataset [63]. These three networks illustrate the state-of-the-art approaches that can be used when sufficient labeled data is present for object detection and classification.

## Section 2.10 Modern Approaches to Shape Classification

The task of classifying objects and shapes has evolved from its onset to much more complicated tasks and innovative applications. As described in Section 2.9, the advent of Deep Learning has revitalized old tasks with new approaches. Fueling this rebirth are advances in applications such as autonomous vehicles and medical object detection. Upon review of older

approaches to shape recognition, a general tradeoff between local specificity and global robustness for features of shapes is apparent. Approaches focusing on local shape descriptions effectively detected occlusions and minor variations, yet they were poor at incorporating the details into the global shape for classification purposes [10, 67, 37, 69]. On the other hand, global descriptors were robust against noise and shape deformations but failed to capture shape details and suffered from occlusions [41, 9, 1, 73]. To solve this issue, many researchers created hierarchical models which incorporated a fusion of local and global features to describe and classify shapes, with progressively better results [49, 63].

An advancement by Xu et al. proposed an invariant, multi-scale shape feature extractor to get the best of both local and global approaches [78]. An algorithm was developed to describe the contours of shapes and combined a variety of classical computer vision techniques such as Gaussian blurs for reducing noise and retrieving shape features such as arc length and area at multiple scales. The algorithm was tested on commonly used datasets, namely the MPEG-7, Articulated, Kimia's 99, and Kimia's 216 datasets. The results of their experiments were promising, showcasing higher accuracy than previous methods on complex shape contours such as animals and objects, with approximately 92% accuracy across the various datasets. This work shows promise in the use of classic computer vision techniques to be used in novel combinations for more robust descriptions and classifications of shapes. Although the retrieval accuracy was high, there remains room for improvement for complex shapes. Regardless, this work inspires the combination of multiple techniques to solve shape matching tasks and showcases the efficacy when properly conducted.

A core challenge in identifying highly complex shapes is when deformations occur as they significantly distort the image. The main deformation that is relevant to this project will be when

overlapping occurs between complex shapes such as fish, seaweed, and bacteria. To account for large deformations due to occlusions, local geometric corrections have been attempted but faced a great hurdle when it came to determining which specific segment of a shape needs the geometric correction. The work by Marvaniya et al. addresses this issue by efficiently searching for the cluster of contour segments that could be potential matches for a geometric correction [47]. They assessed their algorithm on the MPEG-7 dataset and achieved an overall accuracy of approximately 89%, a high score but one that lagged behind others such as Gopalan et al. which had an accuracy of around 93% [29].

The relevance to this study is when they adapted the MPEG-7 dataset to have shapes with 5-15% occlusions. After adapting the dataset, they outperformed Gopalan et al. *b*y 14% and outperformed other similar algorithms by 38%, presenting a strong case for their approach to handling partial occlusions in shapes. This work holds promise as a conceptual framework to handle occlusions, especially for complex shapes.

## Chapter 3: Dataset

The datasets used in this project consist of multiple scientific visual models provided by the ETS organization. Each visual model consists of an assortment of micro and macro 2D shapes. The micro shapes are squares, circles, diamonds, triangles, and arrows. The macro shapes include, but are not limited to, cans, fish, seaweed, bacteria, and water drops.

Three datasets are used for this project, the first of which is the Ocean Water Dataset that consists of 148 visual models. This dataset has human annotations for each of the individual shapes comprising the models, as well as the final LP score for each model. Some errors were detected in the human annotation, but, overall, the dataset can be used as a ground truth comparison for both supervised and unsupervised approaches. Certain methods rely on comparisons with high-quality template images of individual shapes which were initially meant to be provided by the ETS company. Due to unforeseen events with the current CoVID-19 pandemic, these high-quality templates could not be obtained and were instead obtained from four of the 148 images in the Ocean Water Dataset. These four images were therefore excluded from all training and testing, leaving a 144 image Ocean Water Dataset. Furthermore, templates of Diamonds proved difficult to obtain and were thus neglected for template-based approaches. Examples of visual models from the Ocean Water dataset can be found in Figure 3.1.

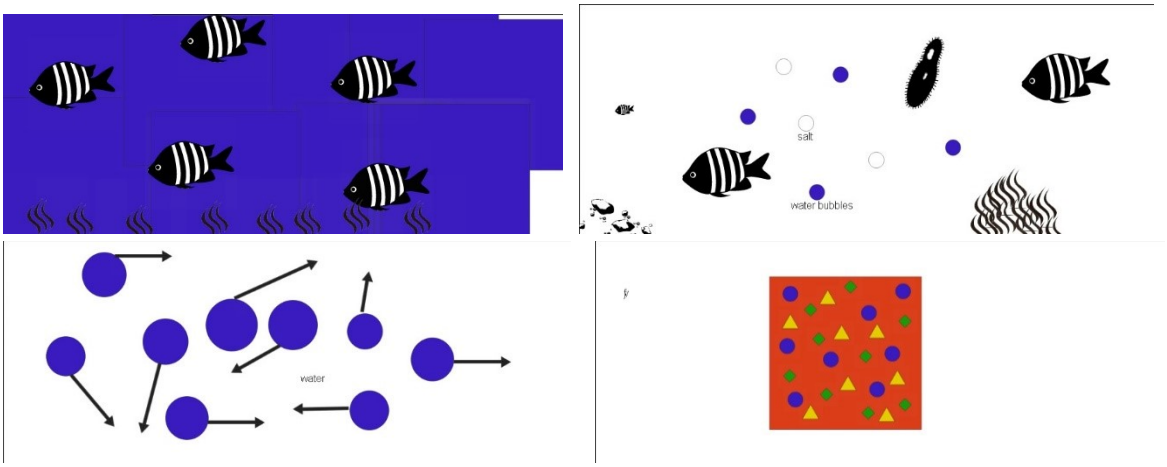



Figure 3.1: Visual models from the Ocean Water dataset

In addition, there are two datasets used, namely the Synthesized Ocean Water and the Two Can datasets. The Synthesized Ocean Water dataset has 30 visual models which are all at LP level 4. However, it lacks labels for the individual shapes. The Two Can dataset consists of 195 visual models ranging from LP levels 1-4, but it too lacks labels for individual shapes. Furthermore, the Two Can dataset contains hand-drawn objects and words, which this project is not designed to recognize. Examples of the Synthesized Ocean Water and Two Can datasets can be found in Figures 3.2 and 3.3 respectively.

The 144 image Ocean Water dataset can therefore be used to evaluate our shape detection algorithms and framework. This dataset can then be combined with the Synthesized Ocean Water Dataset to form a 174 image dataset that can be used to test the LP score of each model. Similarly, the Two Can Dataset can be used to evaluate LP levels. From the quality of the LP scores, correlations to the performance of the object and feature classification framework can be drawn.

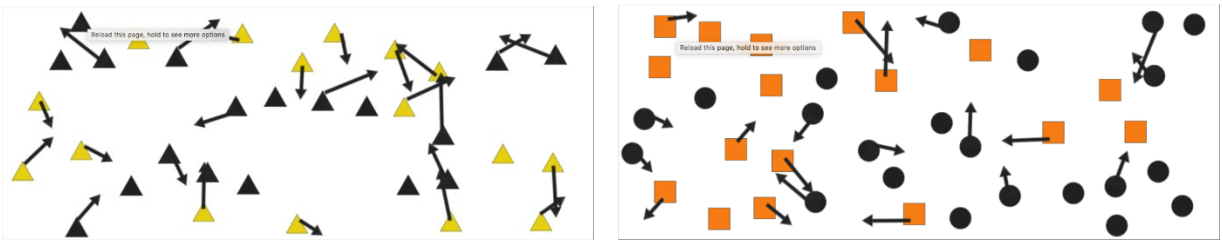


Figure 3.2: Visual models from the Synthesized Ocean Water dataset

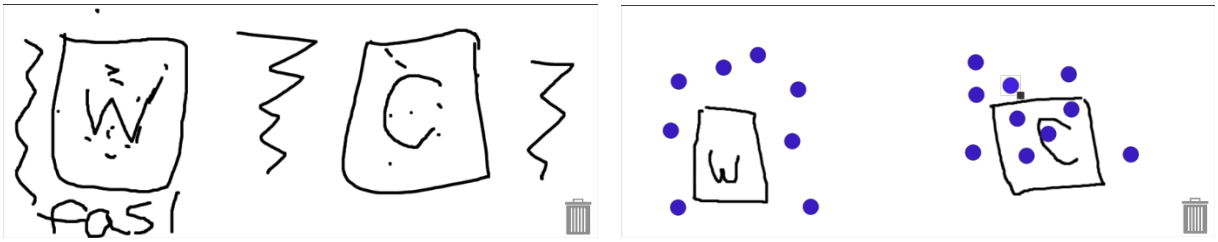


Figure 3.3: Visual models from the Two Can dataset

## Chapter 4: Methodology

The goal of this project is to accurately recognize various shapes in 2D visual models and extrapolate Learning Progression (LP) scores from the objects' classifications and features. In order to achieve this goal, multiple computer vision techniques are integrated, which can be summarized by the following and shown in Figure 4.1. First, template shapes are extracted corresponding to each possible shape except Diamonds due to issues with extraction. Then, the template shapes as well as the visual models undergo color segmentation, are converted to grayscale, and then binarized. Following this, the closed contours are detected for each shape in the visual models and the template images. Following this, the shapes are classified using a cascaded voting system based on a variety of methods including a template independent rules-based approach, feature comparisons with the templates, and a supervised machine learning approach. Finally, the shape classification and features of the shape are used in an automatic grading system which assesses the LP level of the visual model in order to evaluate the students' performance.

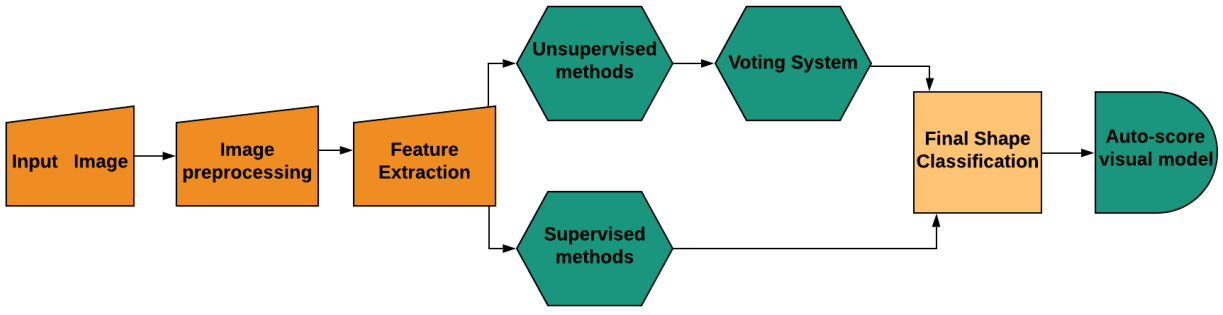



Figure 4.1: Overview of methodology

## Section 4.1 Template Extraction

The first step in this project was to create a subset of template images. These templates are examples of each shape and were screenshots of the individual shapes from four of the images in the 148 Ocean Water Dataset. As described in Chapter 3, these four images were consequently excluded from all further training and testing. Notably, templates of Diamonds could not be extracted, resulting in Diamonds not being classified in template-based approaches including Hu Moment Classification, Area Ratio Classification, Template Matching, and Decision Tree Classification. Examples of the template images can be found below in Figure 4.2.

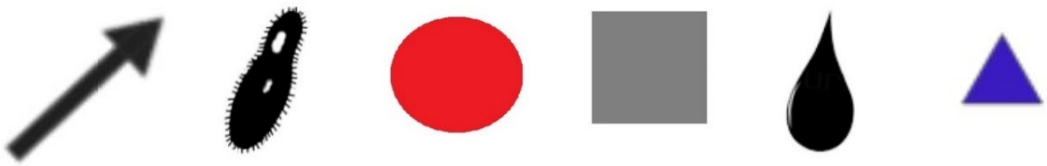


Figure 4.2: Examples of template images

## Section 4.2 Gaussian Blur

A Gaussian Blur is a computer vision preprocessing method which convolves a Gaussian function as a 2D matrix over an image. The goal of a Gaussian Blur is to smooth the colors in an image so that each pixel is impacted by the surrounding pixel values. Essentially, this will improve the signal to noise ratio in an image so that random artifacts and low quality images can better be analyzed. The visual models are of high quality and have little noise so a Gaussian blur was not used on them. On the other hand, the extracted templates had a high variability in quality so a Gaussian blur was applied to them. A Gaussian blur in 2 dimensions mathematically works according to the below function which slides across an image and reiterates at each new pixel.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.1}$$

To illustrate the idea of applying a Gaussian function as a 2D matrix, a representation of a 3 by 3 matrix is shown in Figure 4.3 in which A is the original matrix of pixel values and B is the resulting matrix after the Gaussian function is applied. The pixel itself is represented by the center value of the matrix, in the example below shown by the value 4. Then the surrounding pixels' RGB values are multiplied by the corresponding weights. As the size of the matrix increases the appeared blurriness increases since each pixel is affected by more of its neighboring pixels. In Figure 2.3 an illustration is shown comparing an original image to a 3x3, 5x5, and 7x7 Gaussian blur.

$$B = A * \frac{1}{16} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 4.3: 3 by 3 Gaussian blur matrix

## Section 4.3 Color Segmentation

Prior to extracting features and comparing templates to candidate shapes, both images and templates undergo color segmentation. Color segmentation is a method that refines the colors in an image so that the distcontrast between segments is starker. Colors for each pixel are represented in the Red, Green, Blue (RGB) color space. Mathematically, the pixel values are calculated as a tuple with three values corresponding to the respective RGB channels and each channel ranging from a value of 0 to 255. For example, a tuple of (0,0,0) represents all black and a tuple of (255, 255, 255) represents all white. As the color segmentation is applied identically to templates and visual models, they will both be referred to as images for this section.

The first step in segmenting the images by color is to count the number of pixels that are the same color. A filter was then applied to remove all colors that appeared less than 200 times per

image in order to reduce the noise in the image. Having filtered out low frequency colors, the next step is to apply a window that joins the pixels that are close in value to specify a limited list of colors and further increase the contrast. To determine the closeness of colors, the difference between RGB values in the remaining pixels is calculated. If the absolute difference is determined to be less than 40, then all but one of the colors is consequently removed leaving one color to represent a small range of similar colors. This number was determined experimentally using our dataset. The next step is to create a mask to cover up the image between a set color range. This is accomplished by separating the RGB color space into the three color channels individually. Then, a minimum and maximum value with a difference of 40 is determined for each color channel. Finally, the channels are recombined so that a tuple of minimum and maximum color levels are present. The mask adopts these values and covers every pixel in the image that is not found within this range. Finally, a bitwise_and operator is used to return an image that returns True values, which in this context are pixels that are not covered by the mask. Having segmented the images by color, the final steps in preparing the images to be separated into individual shapes are converting the image to grayscale, followed by inverse-binarizing the image.

## Section 4.4 Contour Detection and Shape Approximation

Contour detection is the process of determining where a curve of continuous pixels with similar color and intensity is within an image. With the color segmentation process complete, the process of detecting contours is greatly simplified. The Suzuki algorithm [74] is implemented which takes a binarized image as input and outputs a list of all the closed-loop contours. The Suzuki algorithm accomplishes this by starting at the corner of an image and then scanning from left to right until a non-zero pixel is detected. It then checks neighboring pixel values to determine if they

are part of the contour (value of one) or outside the contour (value of zero). This process is recursively applied until a closed loop of non-zero value pixels are connected.

The following step is to analyze each closed contour individually and determine the polygon it is most similar to. This approximation is accomplished using the Ramer-Douglas-Peucker (RDP) [20] algorithm. The RDP algorithm takes a contour as its input and reduces the number of points by approximating lines between a series of connected points. In order to approximate the lines, two points are initially chosen and a line is drawn between them. A parameter, epsilon, is chosen which sets the distance from the line drawn. If another point along the contour has a distance value greater than epsilon between the drawn line and itself, then the line is split into two with that new point now connecting both. This process is recursively applied until an approximate polygon composed of multiple straight lines is produced. A tradeoff emerges when deciding on the epsilon value between increased specificity with a low epsilon value and simplifying the contour with a large epsilon. For this project, an experimentally derived epsilon value of 1% of the contour's perimeter is chosen for each contour. The benefit of this approximation is that the number of edges or corner points in a contour can be determined, a useful feature which will be exploited in the following sections.

To gain more information down the line, two bounding boxes and an enclosing ellipse are created around the contour, as shwon in Figure 4.4. The first bounding box simply puts a rectangle around the contour with the sides of the bounding box parallel to the frame of the image. The second is a minimum area bounding rectangle which encapsulates the contour with the smallest possible rectangle that will still cover the entirety of the contour. The added benefit of the minimum Area Rectangle is that the rotation can be calculated within a range of 0 to 90 degrees. The rotation is calculated by first taking the vertically lowest point of the box as the first vertex.

Then, the next three vertices are labelled in a clockwise fashion. The angle between the first and fourth vertex is then calculated counter-clockwise relative to the horizontal, giving an angle of 0 if the vertices are on the same horizontal line and 90 if on the same vertical line. If the angle exceeds 90, a new vertex is chosen as the first vertex and the cycle repeats, thus never having an angle greater than 90.

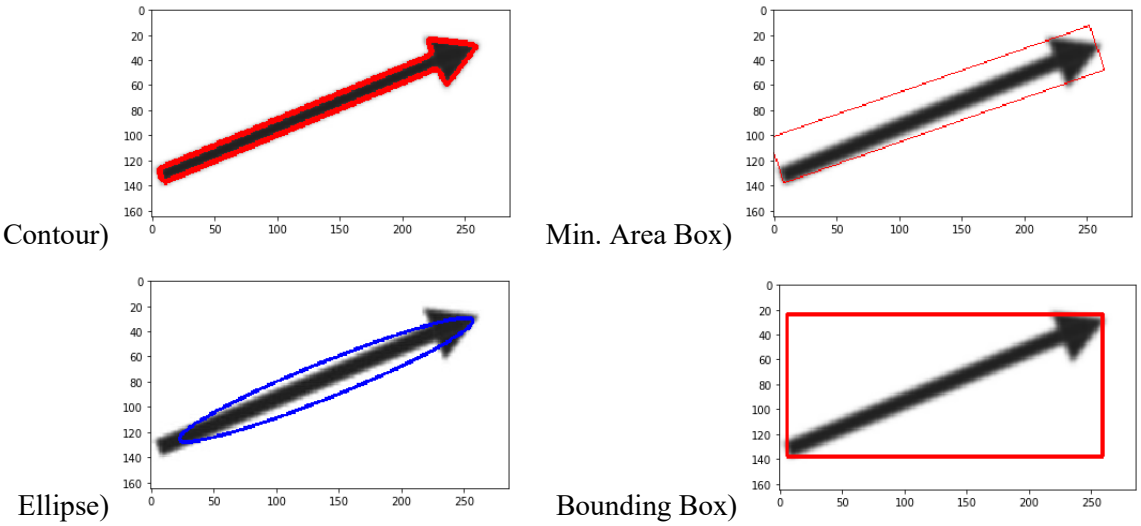



Figure 4.4: Contour and bounding shape examples

## Section 4.5 Rules-based Shape Classification

To classify the detected objects as shapes, many approaches are used. The first is a rule-based classifier which uses the number of edges returned by the approximation of the shapes with the height, width, and rotation returned from the Minimum Area Bounding Box. The rest of this section lists the shape followed by the rules we designed that must be met to classify it as such.

1. Triangle: the number of edges must equal 3 or 5, the height cannot equal the width, and the height should be within a range of +/- 10 pixels of the width.

2. Diamond: the number of edges must equal 4, the rotation of the bounding box must be between 40 and 50 degrees, and the height of the box must be within a range of +/- 2 of the width

3. Square: the number of edges must equal 4, the classifier must not have already classified this contour as a diamond, and the height should be within a range of ± 2 of the width.

4. Circle: the number of edges must be greater than or equal to 7 and the height must be within a range of ±1 of the width.

5. Arrow: the number of edges must be between 7 and 10, the color is black, and the height is not equal to the width.

6. Other: Anything that does not satisfy the above rules is classified as Other

## Section 4.6 Hu Moment Classification

In addition to the rule-based approach to object classification, Hu Moments are utilized to improve prediction accuracy. Hu Moments are seven values that describe a shape in a scale, rotation, translation, and reflection invariant manner. These seven values are calculated from Equations 4.2-4.8 in which the first two are translation invariant, the 3rd and 4th are scale invariant, the 5th and 6th are rotation invariant, and the 7th is reflection invariant. The equations below have M1 to M7 representing the seven values and $i, j$ is the image moment raised to the $i$th and $j$th orders. By undergoing various linear algebraic transformations as described in [32], these invariant properties are achieved. These seven Hu moments are calculated for the outermost contour in a template image and for each contour in the visual model. To determine which shape the detected

contour in the visual model is, distance measures are taken between the Hu Moments of the visual model contour and all the template image contours. Formulas D1, D2, and D3 in Equations 4.9-4.11 accomplish this where $H_i$ is the Hu value for the template (A) and the target contour (B).The trend is that the shorter the distance between 2 shapes is, the more likely they are the same shape.

$$M_1 = (\eta_{20} + \eta_{02}) \tag{4.2}$$

$$M_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{4.3}$$

$$M_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{4.4}$$

$$M_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{4.5}$$

$$M_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})$$
$$[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{4.6}$$

$$M_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{4.7}$$

$$M_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} + 3\eta_{12})(\eta_{21} + \eta_{03})$$
$$[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{4.8}$$

$$D_1(A, B) = \left|\frac{1}{H_i^B} - \frac{1}{H_i^A}\right| \tag{4.9}$$

$$D_2(A, B) = \left|H_i^B - H_i^A\right| \tag{4.10}$$

$$D_3(A, B) = \frac{|H_i^B - H_i^A|}{H_i^A} \tag{4.11}$$

## Section 4.7 Area Ratio Classification

The implementation of three Area Ratios is another method used to classify detected objects. The three used in this project are the ratios between the detected contour's area and the area of three enclosing shapes. As a recap, each binarized contour in the visual models and the templates gets a Bounding Box, a Minimum Area Bounding Box, and a Minimum Area Ellipse to surround it, as shown in Figure 4.4. Additionally, the contour area of each contour is the sum of white pixels present. The areas of the Bounding Box and Minimum Area Bounding Box were then calculated by multiplying the length and width of each box. The area of the Minimum Area Ellipse was calculated by multiplying the major and minor arc lengths by Pi.

Dividing these three areas by the contour area gives a ratio for each contour. This process is repeated for each template, resulting in template and target contour area ratios. The respective ratios of the templates and detected contours can then be compared using the distance operator shown in Equation 4.12. This distance measure is the absolute value of the difference between the template area ratio ($p$) and the visual model's contour area ratio ($q$). Similar to Hu Moments, the smaller the difference value is, the higher the probability of a match.

$$D = | p - q | \tag{4.12}$$

## Section 4.8 Template Matching

Template matching is another method performed on each visual model to classify objects and obtain their location. Template matching is the process of sliding a grayscale template image across a larger grayscale target image. The difference between the portion of the target within the sliding window and the template are calculated and tested against a threshold level. If the matching

percentage is greater than the user specified threshold, then a match is detected and a bounding box is created around the matched object, as shown in Figure 4.5 for the template of Rocks. This bounding box provides the coordinates of the top right and bottom left corners.

The implementation of template matching is accomplished as follows. Having extracted template images for Hu Moment and Area Ratio object classification, these same templates are reused. For this project, the threshold level is .95 which corresponds to a 95% match between the target and the template, which was determined experimentally. The similarity between a template (*T*) and each image (*I)* at points (*x,y*) was calculated by using the normalized cross coefficient as shown in Equation 4.13.

$$R(x,y) = \frac{\sum_{x\prime,y\prime} T(x',y') * I\prime(x+x',y+y'))}{\sqrt{\sum_{x\prime,y\prime} T\prime(x',y') * \sum_{x\prime,y\prime} I\prime(x+x',y+y')^2}} \tag{4.13}$$

Where

$$T'(x',y') = T(x',y') - \frac{1}{(w*h)} * \sum_{x\prime\prime,y\prime\prime} T(x'',y'') \tag{4.14}$$

$$I'(x+x',y+\ y') = I(x+x',y+y') - \frac{1}{(w*h)} * \sum_{x\prime\prime,y\prime\prime} I(x+x'',y+y'') \tag{4.15}$$

An issue with this method is that the visual models have objects in a variety of sizes. Template matching in general is sensitive to scale, which would result in many false negatives if directly applied. To address this problem, the template image is iteratively resized in 10% intervals ranging from 10% of the original size to 200%. At each scale, the matching method is implemented, thus making template matching scale invariant. Having detected objects at multiple scales, the final step is removing duplicate shapes. For example, if a circle is detected at a large scale, a concentric, smaller circle may be detected at the same location. To address this, all shapes

are checked to determine if identically classified shapes overlap in location and, if so, the smaller one of them is filtered out.

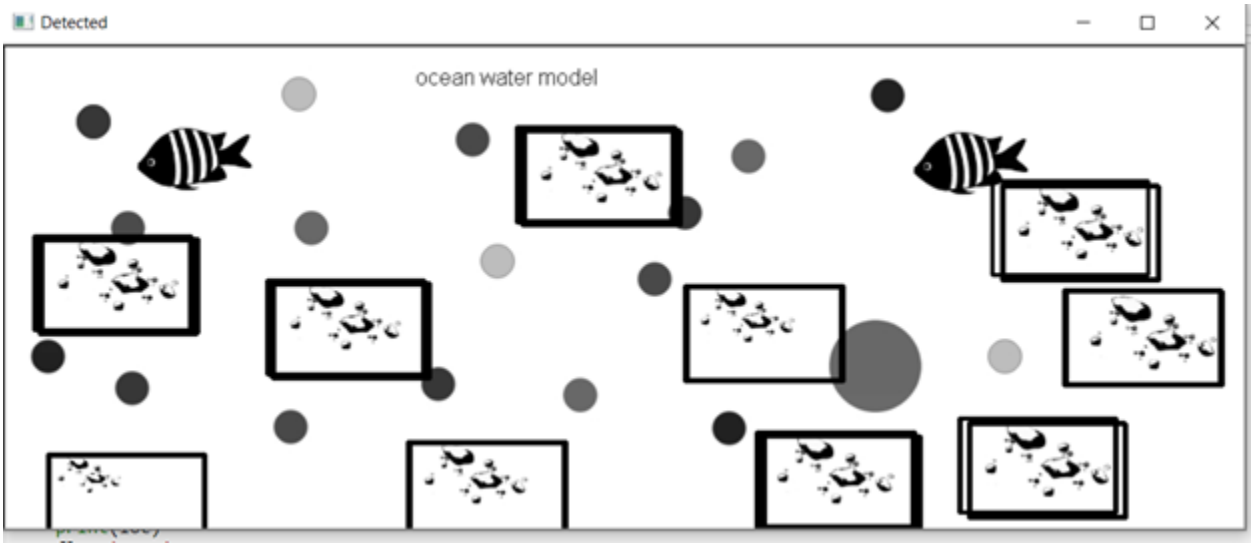



Figure 4.5: Template Matching for rocks with bounding box shown

## Section 4.9 Machine Learning Classification

Machine learning is a family of powerful computational algorithms that has been shown to significantly improve the accuracy of classifying objects in many research studies. Drawing inspiration from these studies and in order to test the performance of a supervised approaches as well, we have used a combination of feature extraction and a machine learning classifier. Since the amount of labeled, ground truth data is very limited for this project as we only used a very limited number of templates for training, only the Decision Tree classifier is currently explored. This will be used as a basis for future machine learning applications as more labeled data is collected.

The Decision Tree classifier operates by taking a feature vector with a corresponding classification label as its training input. It then receives the testing feature vectors, which have identical attributes with different values, but does not take the test sets labels. The features are then ranked in terms of their information gain and used to predict the classification labels in the test set.

For the purposes of this thesis, the final classifications will be the object type, which are extracted from the same templates used for Hu Moments and Area Ratios, are as follows:

List of Features used for Decision Tree:

1. Number of edges in the contour's approximate polygon
2. Ratio of Bounding Box Area to Contour Area
3. Ratio of Minimum Area Bounding Box to Contour Area
4. Ratio of Bounding Ellipse Area to Contour Area
5. Minimum Area Bounding Box rotation
6. Bounding Ellipse rotation
7. Distance between the geometric center and the centroid

## Section 4.10 Cascaded Voting System

To reach the final classification of a shape using the previously described methods, a cascaded voting system is implemented on three levels. The first is for the individual Area Ratios, the second for the three Hu Moments and three Area Ratios, and the last level produces the final object classification.

The three Area Ratios, namely the Bounding Box Ratio, Minimum Area Ratio, and Ellipse Ratio, each finds the most similar contour area to the respective bounding shape ratio in the templates as described in Section 4.7. The similarity measures are ordered as a list and are then set to a majority rules voting system for the top 1, 3, and 5 results. For example, the Bounding Box ratio can produce a list of results in ascending order of similarity such as [Square, Circle, Circle, Square, Square]. If a top 1 voting system is implemented, then the most similar measure is used,

in this case a Square. If a top 3 voting system is used, then the majority of the first three will be the decision, which is a Circle since it has two out of the three best. For a top 5 voting system, the majority of the five highest similarity scores will be used, which is a Square in this case since 3 out of 5 are Squares.

At the next level, the three Hu Moment measures and three Area Ratios each return n independent classifications. Focusing on the Hu Moments, the three results will be used in a majority rules voting system where if any two of the methods produce the same classification, this will become the final Hu Moment classification. The same process is implemented for the three Area Ratios. Thus from the three independent Hu Moment and Area Ratio classifications, only a single Hu Moment and a single Area Ratio classification are produced.

The final voting system combines the Hu Moment and Area Ratio classifications with the Rules-based classification. Another majority-rules voting system is implemented between these three. For example, if the Rules-based and the Hu Moment classifications match, then the final classification of the object is whatever these two methods predicted. A diagram of this three-level cascaded voting system is illustrated in Figure 4.6.

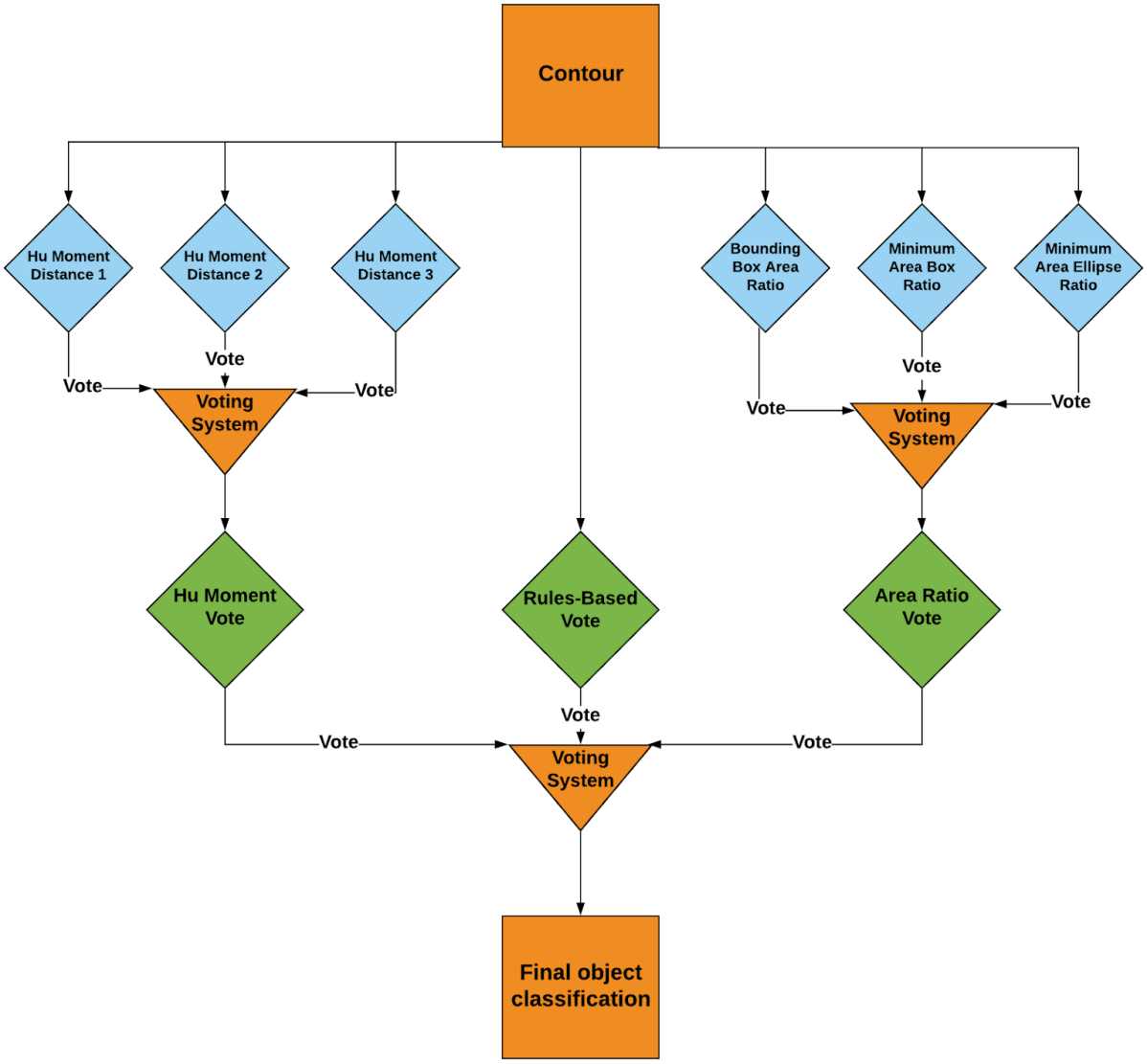



Figure 4.6: Flow Chart of the cascaded voting system

A post-processing step checks that multiple, identical shapes are not detected at the same exact location. If they are, then priority is given to one instance of the shapes while the rest are deleted so that duplicates are not predicted. Finally, what began as a contour detected in a visual model is classified as a shape along with its characteristics such as orientation, location, color, width, and height.

## Section 4.11 Arrow Orientation

The determination of arrow orientation is a feature of particular importance to ETS as thesy portray information about the behavioral movement of objects. To our knowledge, an effective method relevant to this context does not exist so we propose a novel approach to determine arrow orientation. The first step in calculating the orientation was to take the contour of each candidate arrow in an image and fit a rectangular Bounding Box, a minimum Area Bounding Box, and a minimum Area Ellipse around it. Each of these has center coordinates that can slightly vary based on the angle of the arrow, so the average of all three was calculated and used as the geometric center of the arrow. Then, the centroid of the arrow was derived to determine the coordinates of the arrow's center of mass.

$$M_{i\,j} = \sum_x \sum_y x^i y^j * I(x,y) \tag{4.16}$$

$$\bar{x} = \frac{M_{1\,0}}{M_{0\,0}} \tag{4.17}$$

$$\bar{y} = \frac{M_{0\,1}}{M_{0\,0}} \tag{4.18}$$

$$Centroid = (\bar{x}, \bar{y}) \tag{4.19}$$

Equation 4.16 is the general formula for deriving the moments of an image. $M_{i\,j}$ denotes the *i*th and *j*th order moment of an image, *x* and *y* denote the coordinates within an image, and *I(x,y)* represents the pixel intsenity at location *(x,y)*. Since the images have been binarized, the intensity will always be one for white pixels and zero for black pixels. Therefore, the $0^{th}$ order moment $(M_{0\,0})$ is the sum of all white pixels, i.e. the area of the contour. The centroid is then calculated as the first order of each coordinate normalized by the contour area, resulting in the center of mass of the shape.

Equipped with the geometric center and the centroid, the next step was to use both of these to determine the direction of the arrow head. The overarching idea that guided this approach was that the centroid would be shifted in a certain direction relative to the geometric center in the direction of the arrow head. An analogy would be to picture a regular, rectangular see-saw. In this case, the geometric center and the center of mass are equal. Replacing the rectangular see-saw with an arrow while keeping the geometric center the same, the see-saw will tilt in the direction of the arrow head because the center of mass is shifted in that direction. Using this insight, a rules based approach was created to decide the angle of each arrow.

Initially, the angle of each arrow is calculated between 0-180 degrees relative to the vertical based on the minimum Area Ellipse. For consistency, this angle will be called the ellipse angle for the remainder of this section. In this scenario, an arrow facing North and an arrow facing South both have angles of rotation equal to zero degrees. Three sets of rules were created to adjust for this. The first set, listed below, determines the angle for arrows in 90 degree intervals facing North, South, East, and West based on a three degree margin of error from the ellipse angle. An example of the implementation of these rules is found in Figure 4.7.

1. If the ellipse angle is greater than 177 degrees or less than 3 degrees and the Y coordinate of the centroid is higher than the geometric center, then the angle equals 0 representing a North facing arrow.

2. If the ellipse angle is greater than 177 degrees or less than 3 degrees and the Y coordinate of the centroid is lower than the geometric center, then the angle equals 180 representing a South facing arrow.

3. If the ellipse angle is between 87 and 93 degrees and the X coordinate of the centroid is to the right of the geometric center, then the angle equals 90 representing an East facing arrow.

4. If the ellipse angle is between 87 and 93 degrees and the X coordinate of the centroid is to the left of the geometric center, then the angle equals 270 representing a West facing arrow.

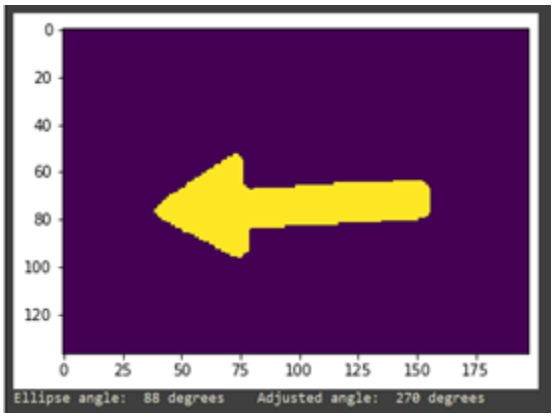


Figure 4.7: Adjusted arrow orientation based on 3-degree margin of error

The next set of rules once again determine the orientation of arrows that could potentially face North, South, East, or West. To differentiate from the above rules, these accommodate for small differences in the coordinates of the centroid and the geometric center that still correspond to one of the four cardinal directions, as shown in Figure 4.8. The motivation to do a second set of rules specifically for these cardinal directions is due to the higher frequency of errors at these points because the ellipse angle goes from 0-180 causing a greater number of errors at these points. The rules based on coordinate similarity are as follows:

1. If the centroid's X component is within 1 pixel of the geometric X component and the centroid's Y component is higher than the geometric Y component, then the angle equals 0 corresponding to North.

2. If the centroid's X component is within 1 pixel of the geometric X component and the centroid's Y component is lower than the geometric Y component, then the angle equals 180 corresponding to South.

3. If the centroid's Y component is within 1 pixel of the geometric Y component and the centroid's X component is to the right of the geometric X component, then the angle equals 90 corresponding to East.

4. If the centroid's Y component is within 1 pixel of the geometric Y component and the centroid's X component is to the left of the geometric X component, then the angle equals 270 corresponding to West.

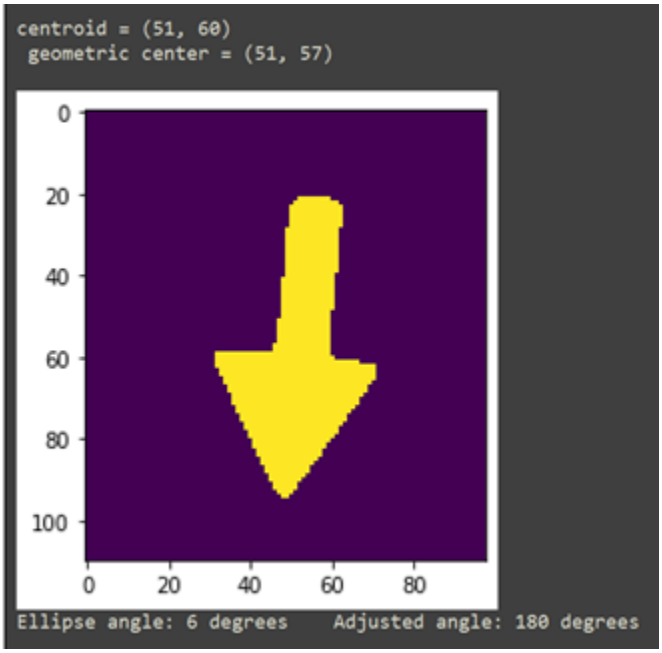


Figure 4.8: Adjusted arrow orientation based on coordinate similarity

Having accounted for the cardinal directions with a three degree margin of error and a one pixel margin of error, the third set of rules determines whether to adjust the intermediate angles by 180 degrees. An example of adding 180 degrees is shown in Figure 4.9.

1. If the centroid is Northeast of the geometric center, then the arrow angle equals the ellipse angle.
2. If the centroid is Southeast of the geometric center, then the arrow angle equals the ellipse angle.
3. If the centroid is Southwest of the geometric center, then the arrow angle equals the ellipse angle plus 180 degrees.
4. If the centroid is Northwest of the geometric center, then the arrow angle equals the ellipse angle plus 180 degrees.

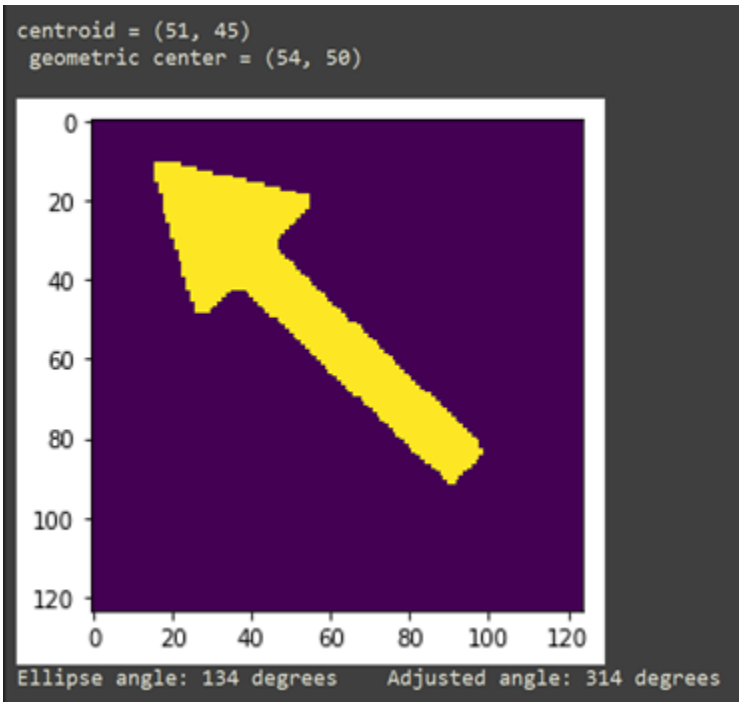


Figure 4.9: Adjusted Northwest arrow orientation

## Section 4.12 Scoring of Visual Models

The evaluation of the visual models consists of three general steps. First, the individual shapes and corresponding features are extracted from each model, as described in Sections 4.2 to 4.10. Then, these features are aggregated to a second level of 10 features that describe the model. The final step is using these aggregated features to predict the Learning Progression (LP). As described in Section 2.1, a LP is a score of 1 (lowest) to 4 (highest) that represents a student's understanding of a concept, in this case the concept of "Matter". The LP used for this study consists of four dimensions each with their own sub-progressions as displayed in Table 4.1, where the combination of dimensional scores maps to the overall LP score. The dimensions are Scale (S), Material Identity (MI), Behavior (B), and Distribution (D). The Scale dimension portrays the composition of Matter based on the micro and macro objects. The Material Identity dimension portrays the number and identity of the objects used in the visual model. The Behavior dimension measures the movement of the objects relative to each other. Finally, the Distribution dimension examines the spatial locations of objects relative to each other.

| | S | | | | MI | | | B | | | | D | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 3 |
| LP-1 | X | X | | | X | X | | X | | | | X | | |
| LP-2 | | X | X | | | X | | X | X | | | X | | |
| LP-3 | | | X | X | | X | X | | X | X | | | X | |
| LP-4 | | | | X | | X | X | | | X | X | | X | X |

Table 4.1: Mapping Learning Progressions to four dimensional sub-progressions

These four dimensions were grouped into two broad feature categories: counting-based and spatial-based features. Counting-based features include the Scale, Material Identity, and Behavior

dimensions. Scale is measured by the ratio of macro and micro objects with each visual model having its own target ratio. Material Identity is evaluated by the difference from the Expected Individual Count (EIC) of microscopic objects, where the EIC is the expected number of specific micro-objects for each question. The Behavioral dimension is interpreted by the orientation and length of arrows where the length denotes the velocity of an object.

Spatial-based features include the Distributive dimension for objects. The k-Nearest Neighbors (k-NN) [17] algorithm is used to determine the relative clustering of particles based on the inter-particle distance. To determine both local and global relations of objects values of k=3 and k=10 were used respectively.

| Features | Description |
|---|---|
| Counting-based | |
| Micro-object types | Number of each type of micro-object |
| Macro-object types | Number of each type of macro-object |
| Micro-object color types | Number of types of micro-object colors |
| EIC deviation | EIC minus count of each type of micro-object |
| Arrows | Count of arrows |
| Arrow lengths | Mean length of arrows |
| Arrow randomness | Variance of arrow orientations |
| Spatial-based | |
| *k*-NN = 3 | Mean distance to 3 nearest micro-bojects |
| *k*-NN = 10 | Mean distance to 10 nearest micro-objects |
| Dispersion | Mean normalized spread of micro-objects |

Table 4.2: Descriptions of 10 features used to score visual models

Once a total of 10 features were extracted as shown in Table 4.2, the final step was to score the visual models and map the results to a LP score. The score prediction process utilized eight regressors and classifiers, as shown in the list below. We employ these various learners in the SciKit-Learn machine learning framework via SKLL for experimentation [79, 80]. The results of these learners are tuned to a Quadratic Weighted Kappa (QWK) score which compares the predicted results to human-annotated LP scores. The range of QWK scores is from 0 to 1, where 0 means a completely random prediction and a score of 1 represents a complete match between predicted and ground truth labels. A benefit of these approaches is that the outputs are human-understandable which is highly beneficial for educational tests.

List of Learners:

1. Linear Regression
2. Logistical Regression
3. Decision Tree Regression
4. Support Vector Regression
5. Random Forest Regression
6. Decision Tree Classifier
7. Support Vector Classification
8. Random Forest Classification

# Chapter 5: Experimental Results

To determine the results of this project, the experiments can be split into two portions. The first of these is the evaluation of correct object recognition within the visual models. The second part is the prediction of Learning Progression (LP) scores, followed by the comparison between these predictions and human annotated LP scores.

## Section 5.1 Object Classification Results

The first part of this project tests multiple approaches to classify shapes and their features in images. These experiments were conducted on the Ocean Water dataset, provided by Education Testing Services (ETS), which had 50 doubly-annotated visual models and an additional 94 single-annotated models. The single-annotated models were prone to more human error for the ground truths, especially for squares, leading to two separate experiments being conducted. The first on the 50 images separately and the second on the combined 144 images.

For both the 50 visual models and the 144 visual models in the Ocean Water Dataset, the same metrics were used for evaluation. The first two metrics are the Dataset Accuracy and Average Accuracy per Image. The Dataset Accuracy is calculated as the total number of True Positive predictions divided by the total number of instances in the ground truth. The Average Accuracy per Image is calculated by first dividing the number of True Positives by the total number of objects per image. Then, the resulting accuracies per image are averaged to produce the final output. The

motivation for using two accuracy measures is that the Dataset Accuracy is less sensitive to outliers in individual model accuracy, whereas the Average Accuracy per Image is more sensitive to outliers. Since Diamonds were not predicted for Hu Moment Classification, Area Ratio Classification, Template Matching, and Decision Tree Classification, Diamonds will be excluded from both accuracy measures for these methods. Furthermore, since the goal of this project focuses on micro-object detection, the category of Other shapes will also be excluded from all accuracy measures.

Aside from the two accuracy measures, the Precision, Recall, and F1 Score are calculated for each object type. The calculations for these three are illustrated below in Equations 5.1-5.5 where $T_P$ is the number of True Positives, $F_P$ is the number of False Positives, and $F_N$ is the number of False Negatives for each object type. These will illustrate the quality of classifications based on each type of object rather than the accuracy of the overall visual model. The Precision, Recall, and F1 measures were calculated for the two best approaches in the accuracy measures.

$$Dataset\ Accuracy = 100 * \frac{T_P}{Total\ \#\ objects} \tag{5.1}$$

$$Average\ Accuracy\ per\ Image = 100 * Avg(\frac{T_P\ per\ image}{\#\ objects\ per\ image}) \tag{5.2}$$

$$Precision = 100 * \frac{T_P}{T_P+F_P} \tag{5.3}$$

$$Recall = 100 * \frac{T_P}{T_P+F_N} \tag{5.4}$$

$$F1\ Score = 2 * \frac{Precison * Recall}{Precision+Recall} \tag{5.5}$$

A variety of approaches were implemented to produce these results. First, the individual methods of the Rules-based Classification, Hu Moment Classification, Area Ratio Classification,

and the Decision Tree Classification are conducted. The Area Ratios method, as described in Section 4.10, has a voting system for the three individual ratios that classify based on a majority of the top *n* results where *n* can equal 1, 3, or 5. The Area Ratios at each of these three voting levels are combined with the Hu Moment and Rules-based votes to form a Cascaded Voting System which is denoted by Voting System (*n* votes), along with all the other methods, in Figures 5.1 and 5.2 below.

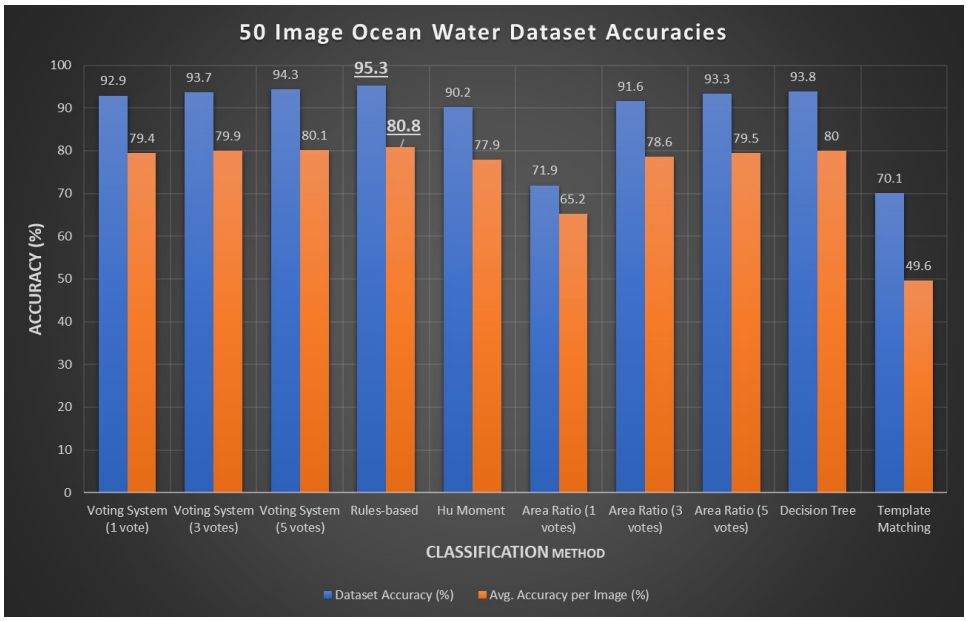



Figure 5.1: Accuracies of classification methods on the 50 image Ocean Water dataset

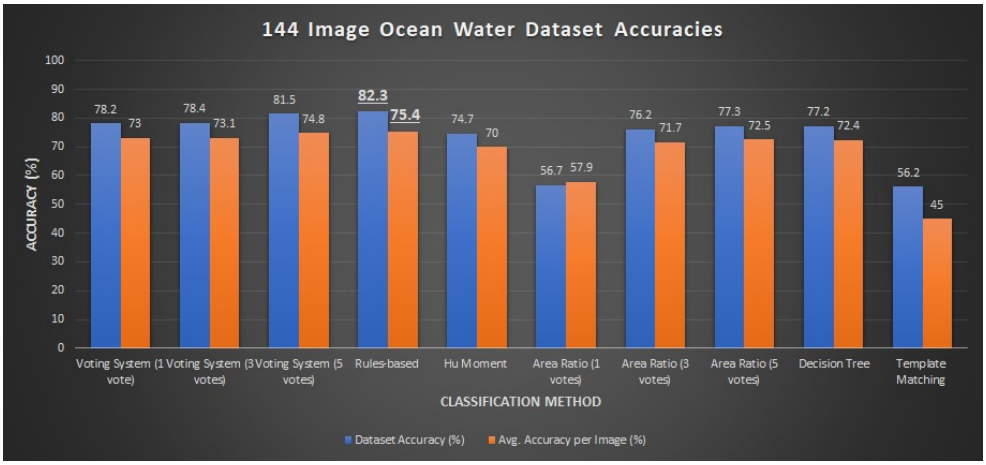



Figure 5.2: Accuracies of classification methods on the 144 image Ocean Water dataset

The results of the Dataset Accuracy and Average Accuracy per Image show that, on both the 50 image and 144 image datasets, the Rules-based Classification method is the best with the 5-vote Voting System right behind it. This implies that the Area Ratios and Hu Moments do not help improve the Rules-based approach in increasing the overall accuracy. On the other end of the spectrum, Template Matching is the worst performer, owing to the poor quality of templates and the inability to rotate templates of arrows. To gain a clearer idea of why the Average Accuracy per Image is drastically lower than the Dataset Accuracy, each individual image's accuracy was analyzed for the Rules-based Classification. It was discovered that images with objects in a non-white background region were getting much lower accuracies than the median and mode scores of 100%. This points to an issue at the color segmentation portion which can be addressed in future work. An example of a problematic visual model is displayed below in Figure 5.3.

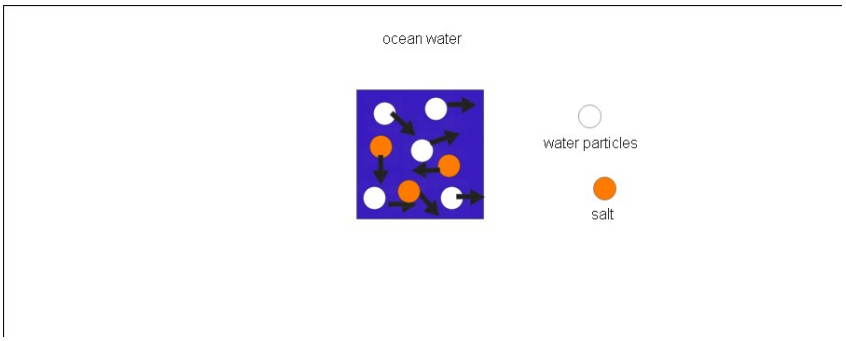



Figure 5.3: Low accuracy visual model due to non-white background

The resulting Precision, Recall, and F1 Scores for the Voting System with 5 votes and the Rules-based Classification are listed below in Tables 5.1 and 5.2. These measures are conducted on both the 50 image Ocean Water dataset and the 144 image dataset. For the 50 image dataset, the Voting System had a higher F1 Score and Precision than the Rules-based Classification for Arrows, indicating that fewer False Positives were predicted. Since Arrow detection was of high importance for ETS, these relatively higher scores for the Voting System validated its use. On the

other hand, the Rules-based Classification had a higher Recall indicating fewer False Negatives were predicted. The rest of the shapes have identical values indicating that the Voting System classified the rest of the shapes in the same manner for the 50 images or that the rule-based approach had a major influence in the voting system. A larger discrepancy emerges on the 144 image dataset. The Rules-based Classification has a higher F1 Score for Arrows, Circles, and Triangles indicating fewer false predictions, while the Voting System achieved higher precision for Squares. Notably, the F1 Scores for Squares are around 20% lower in the 144 image dataset for both methods, matching the observation that the larger dataset contains many ground truth errors for the Squares count because it incorrectly counts background boxes as Squares. Further evidence of the 144 image dataset having mislabeled Squares are the high Precision measures of 98.1% for the Voting System and 95.7% for the Rules-based Classification.

| Voting System (5 Votes) on 50 image Ocean Water dataset | | | | | |
|---|---|---|---|---|---|
| | Arrow | Circle | Square | Triangle | Diamond |
| Precision | 95.7 | 93.5 | 78.6 | 96.0 | 100.0 |
| Recall | 90.7 | 94.7 | 94.3 | 88.9 | 100.0 |
| F1 Score | 93.1 | 94.1 | 85.7 | 92.3 | 100.0 |
| Rules-based Classification on 50 image Ocean Water dataset | | | | | |
| Precision | 86.4 | 93.5 | 78.6 | 96.0 | 100.0 |
| Recall | 97.9 | 94.7 | 94.3 | 88.9 | 100.0 |
| F1 Score | 91.8 | 94.1 | 85.7 | 92.3 | 100.0 |

Table 5.1: Precision, Recall, and F1 Score on the 50 Ocean Water dataset

| Voting System (5 Votes) on 144 image Ocean Water Dataset | | | | | |
|---|---|---|---|---|---|
| | Arrow | Circle | Square | Triangle | Diamond |
| Precision | 90.0 | 97.4 | 98.1 | 84.5 | 100.0 |
| Recall | 78.4 | 87.2 | 49.4 | 92.2 | 100.0 |
| F1 Score | 83.8 | 92.0 | 65.7 | 88.2 | 100.0 |
| Rules-based Classification on 144 image Ocean Water Dataset | | | | | |
| Precision | 90.6 | 97.6 | 95.7 | 94.7 | 100.0 |
| Recall | 84.2 | 87.4 | 49.7 | 92.2 | 100.0 |
| F1 Score | 87.3 | 92.2 | 65.4 | 93.4 | 100.0 |

Table 5.2: Precision, Recall, and F1 Score on the 144 Ocean Water dataset

## Section 5.2 Learning Progression Scoring

The ultimate goal of this project is to automatically score visual models with a Learning Progression (LP) score between 1-4. The LP scores are classified based on the combinations of the 10 aggregated features across four dimensions as described in Section 4.12. The datasets are shuffled into a 10-fold cross validation with all LP scores present in each fold. The predicted LP scores are then compared to human annotated scores by using supervised regressors and classifiers. The results of these are all tuned to Quadratic Weighted Kappa (QWK) which measures the similarity between the predicted and human annotated scores on a scale of 0 to 1, where 0 represents no match and a score of 1 is an identical match. Additionally, Learning Curves of the Linear Regressions are displayed for each dataset, displaying how the error gap between training and validation sets decreases as the number of training examples increases.

The first dataset to be analyzed is the 144 image Ocean Water dataset. To compare the Rules-based Classification and Voting System, both methods were independently used to extract all the necessary features. Then, a 10-fold cross validation was performed. The resulting average QWK scores for each type of learner are displayed in Figure 5.4 and the respective Learning Curves in Figure 5.5. The same methods were applied to the 174 image Ocean Water dataset that includes the 30 Synthesized Ocean Water visual models. The average QWK Score for the larger dataset are displayed below in Figures 5.6 and the respective Learning Curves in Figure 5.7.

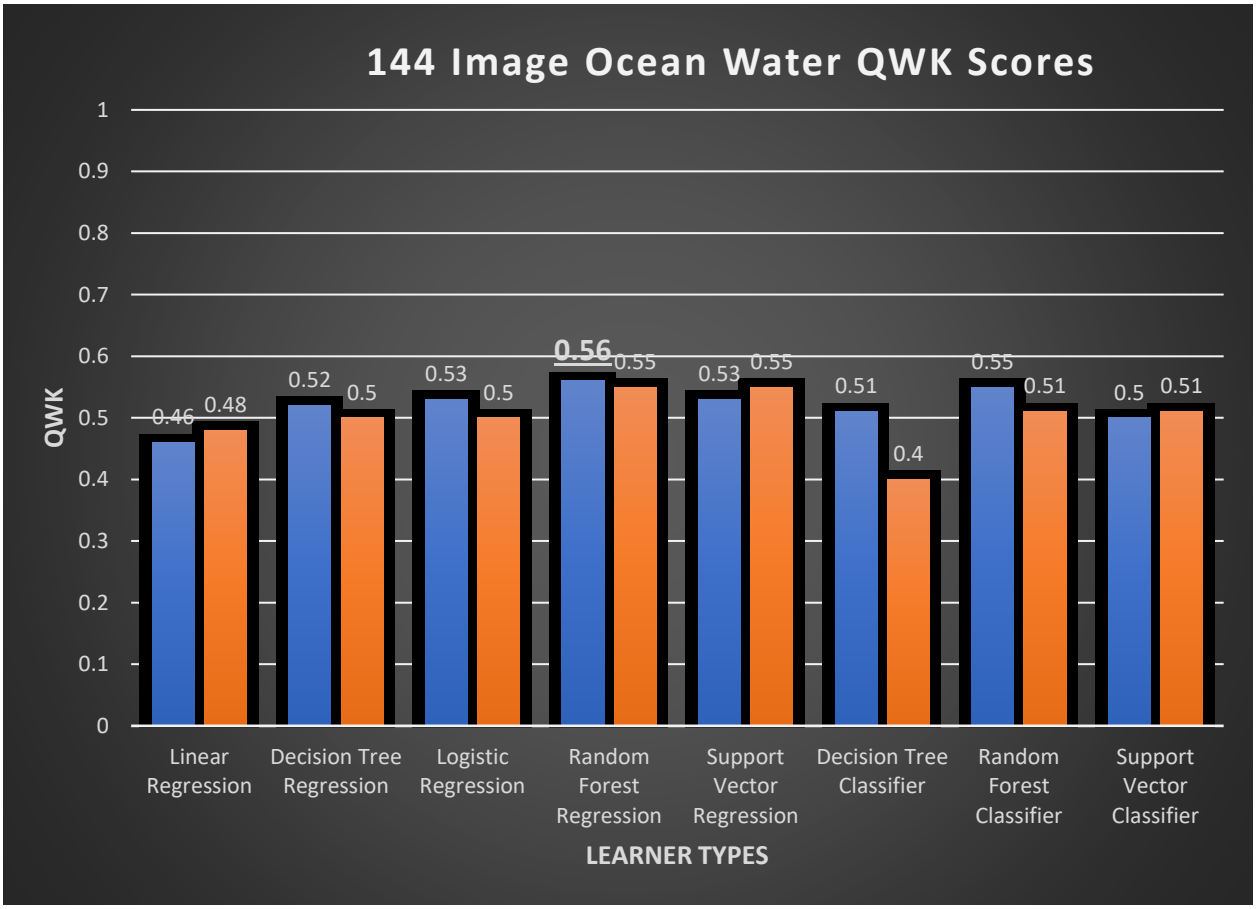
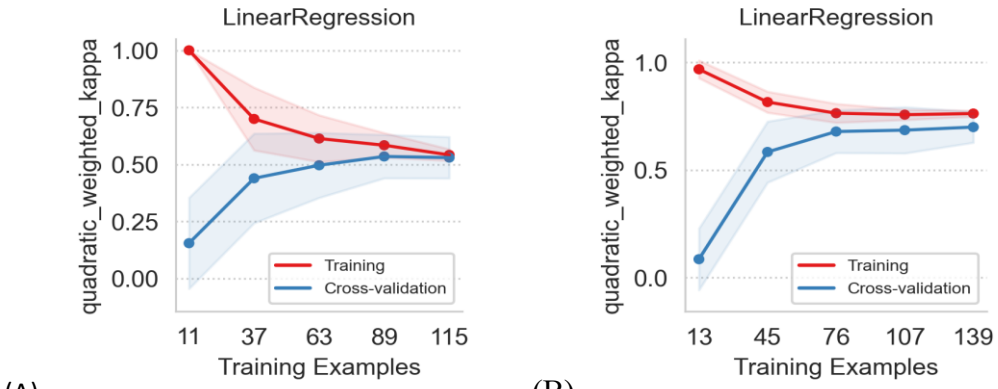


Figure 5.4: QWK Scores for 8 learners on the 144 image Ocean Water dataset




(A) (B)

Figure 5.5: 144 Image Ocean Water learning curves for voting system (A) and rules-based (B) methods

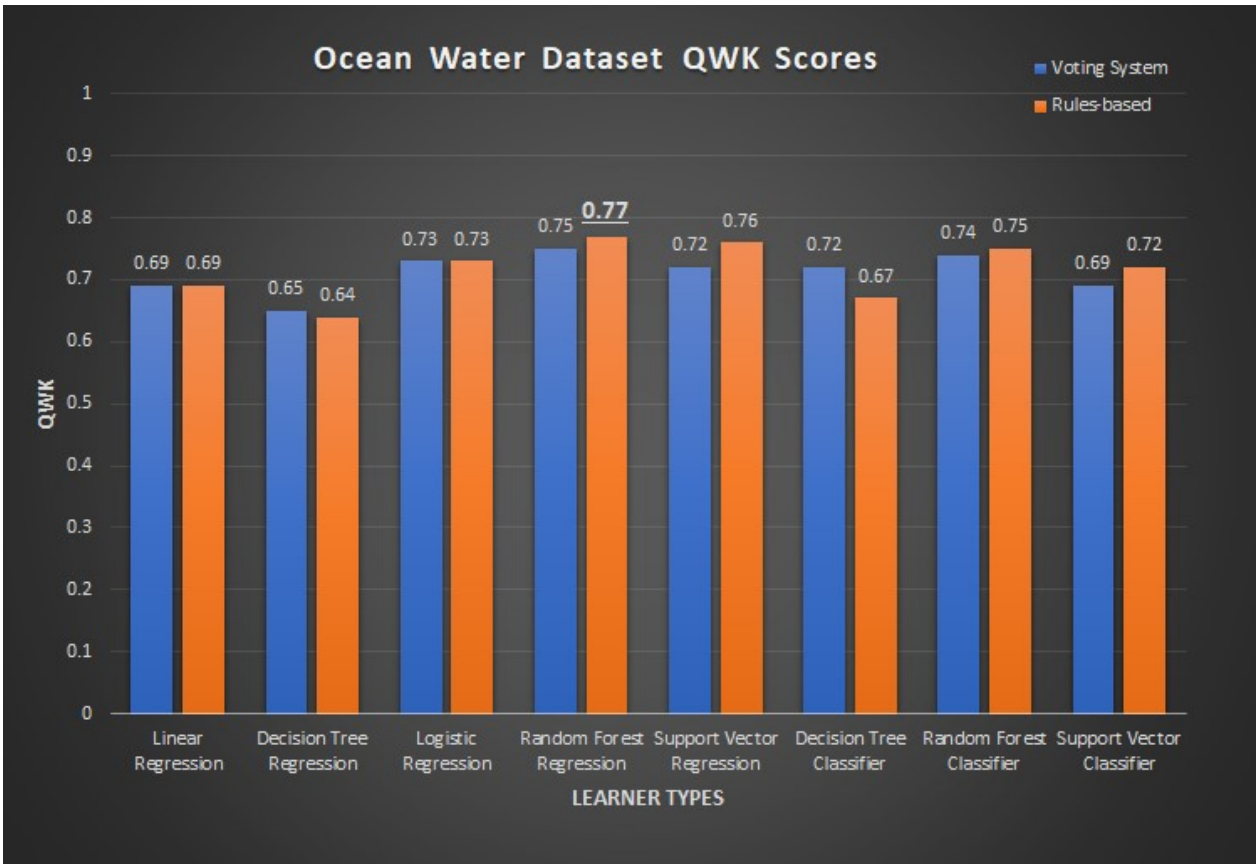
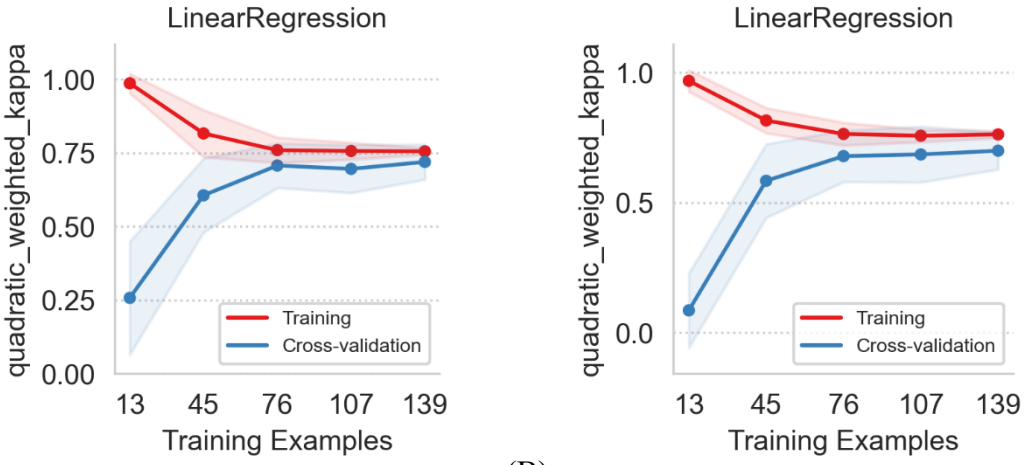



Figure 5.6: QWK Scores for 8 learners on the 174 image Ocean Water dataset




(A) (B)

Figure 5.7: 174 Image Ocean Water learning curves for voting system (A) and rules-based (B)

The results for the 144 Image Ocean Water Datset has a peak Quadratic Weighted Kappa (QWK) score of .56 for the Voting System and .55 for the Rules-based Classificaation, both of which were achieved with the Random Forest Regressor. For the 174 Ocean Water Dataset with the Voting System has an average QWK score of .711 with a standard deviation of .033. The Voting System scored higher for Decision Tree Classification and Regression. On the other hand, the Rules-based Classification has an average QWK score of .716 with a standard deviation of .046. The peak score for both methods was with the Random Forest Regressor, achieving scores of .75 for the Voting System and .77 for the Rules-based Classification. This shows a strong ability to predict the LP level of an Ocean Water instance based on its automatically extracted features. These high scores further validate this project's effectiveness in classifying the types and characteristics of the objects. The Learning Curve in Figure 5.7 also shows the expected pattern of the error minimizing as the QWK scores converge.

The second dataset to be analyzed is the 195 image Ocean Water Dataset. Once again, the features were autoamtically extracted with the Rules-based Classification and Voting System,

followed by a 10-fold cross validation. The average QWK Score of the 10 folds is displayed below in Figure 5.8.

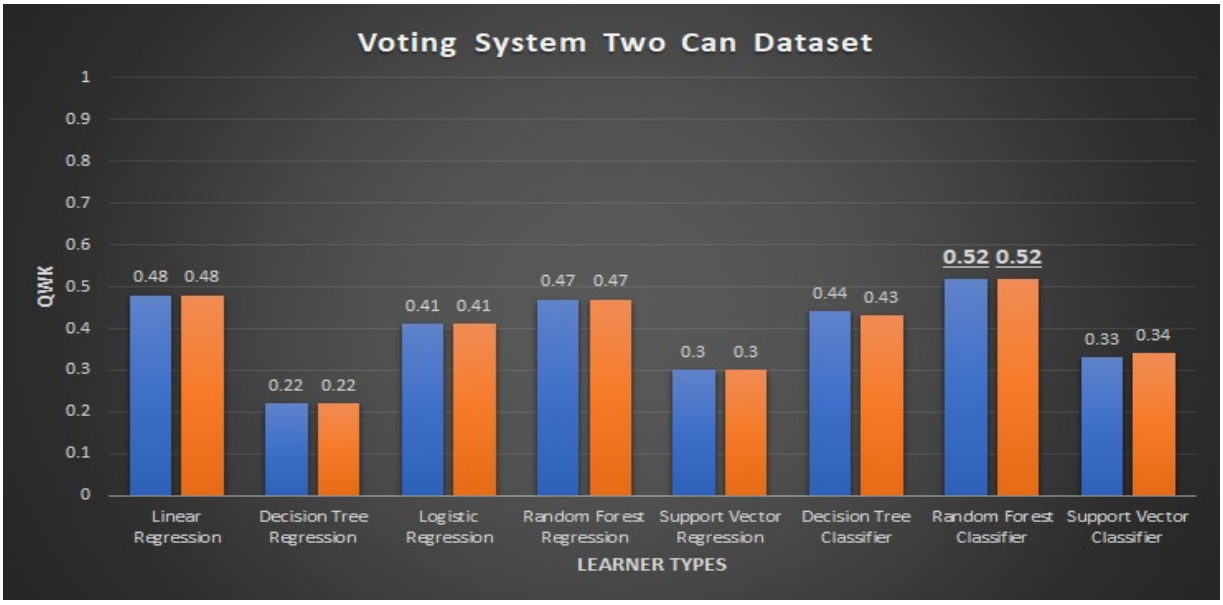
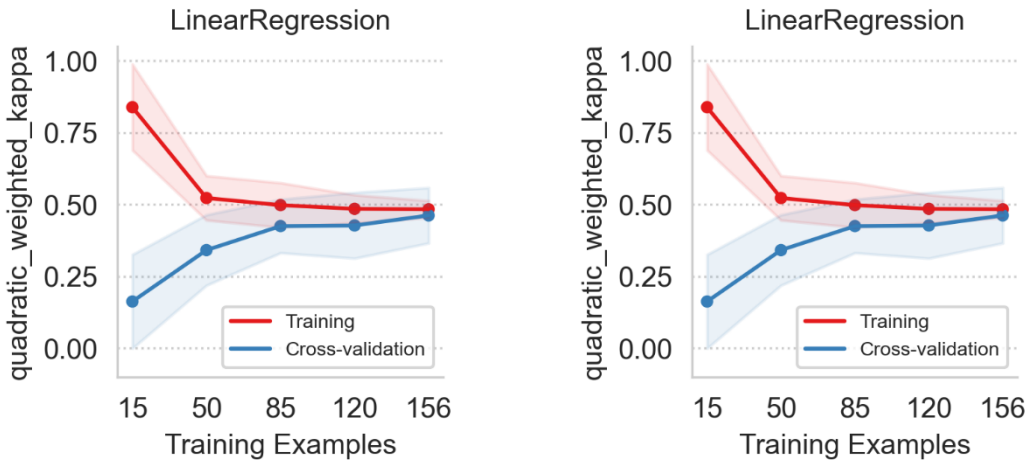



Figure 5.8: QWK Scores for 8 learners on the Two Can dataset




(A) (B)

Figure 5.9: Two Can learning curves for voting system (A) and rules-based (B)

The resulting QWK Scores are nearly identical for both the Voting System and Rules-based Classification approaches, providing greater evidence of their similarity at the fundamental feature extraction level. This further shows that both of them generalize to novel datasets in a similar manner. The average QWK Score for this dataset is .4 with a standard deviation of .1 for both

methods. The peak score is achieved using the Random Forest Classifier with a QWK score of .52 for both methods. These results are relatively lower than those for the Ocean Water Dataset, a finding consistent with the design of this system. The Two Can dataset contains hand drawn objects and labels which our system was not designed to recognize. The Learning Curves in Figure 5.9 for the Linear Regression still shows the error minimizing as the number of training examples increases, but it converges slightly below a QWK score of .5 as compared to the approximate convergence value of .75 for the Ocean Water dataset. This culminates in showing worse LP prediction for the Two Can dataset than the Ocean Water dataset. However, we expect that the results might potentially improve once the new hand drawn objects problem is addressed.

## Section 5.3 Cross Domain Analysis

After testing the individual datasets, a cross domain analysis was performed between the Ocean Water and Two Can datasets. Figure 5.8 below illustrates the results of the first of these analyses in which the Ocean Water dataset is used for training and the Two Can dataset is used for testing. The results show a peak score of .37 using Support Vector Regression. The average QWK score between the eight learners is .28 with a standard deviation of .065. These results are consistent with the theoretical expectations that cross domain scores will be lower than when testing on the same dataset as training.

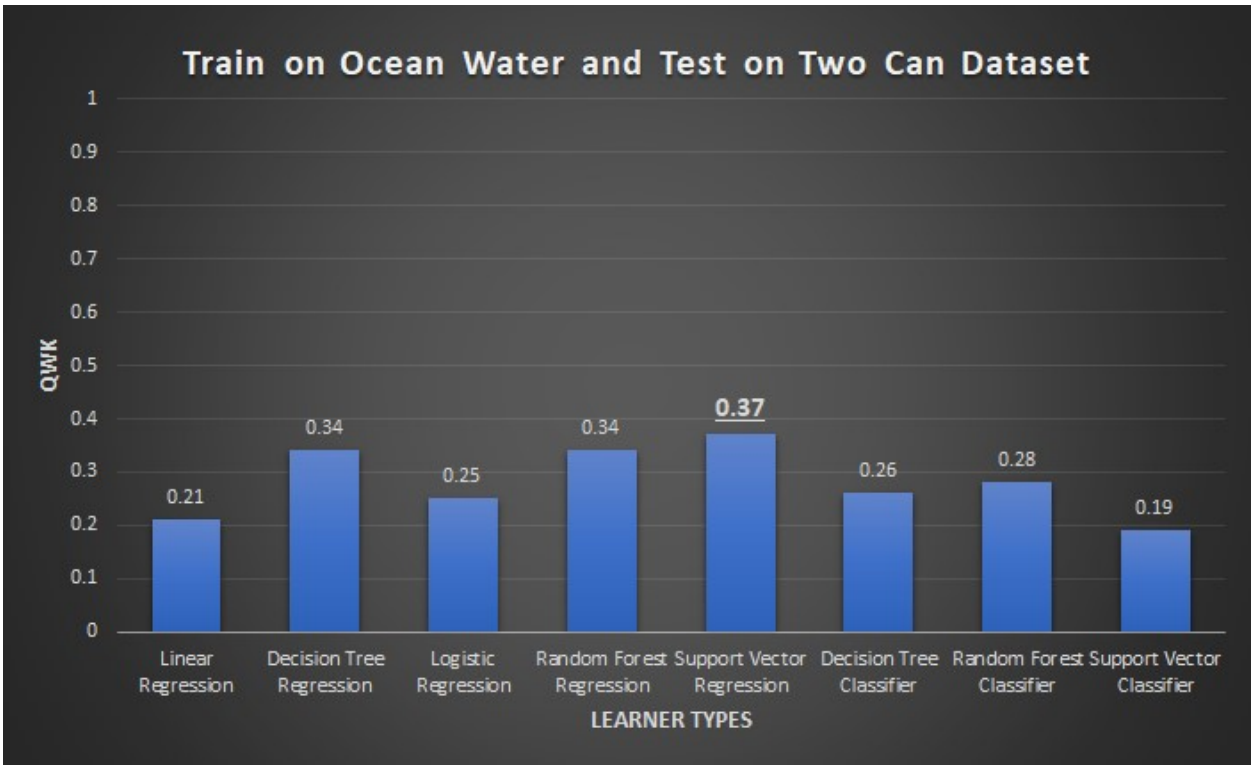


Figure 5.10: Cross domain QWK trained on Ocean Water dataset and tested on Two Can dataset

The second cross domain analysis is the reverse of the first. In this case, the Two Can dataset is used for training and the Ocean Water dataset is used for testing. Notably, there are 32 instances in the Ocean Water dataset that have an LP score of 4, whereas there are no instances of an LP score of 4 in the Two Can dataset. If these instances remained in the Ocean Water dataset, it would be impossible to predict them as LP 4 based on the current training data. For this reason, the 32 instances were removed, leaving a 142 image Ocean Water test set. As displayed in Figure 5.9, the results show a peak QWK score of .49 for Logistic Regression, an average score of .27 for the eight learners, and a standard deviation of .137. These results are also consistent with the expectation that the cross domain analysis yields lower scores than when 10-fold cross validation was applied directly on the Two Can dataset, i.e., learning from the same exact domain.

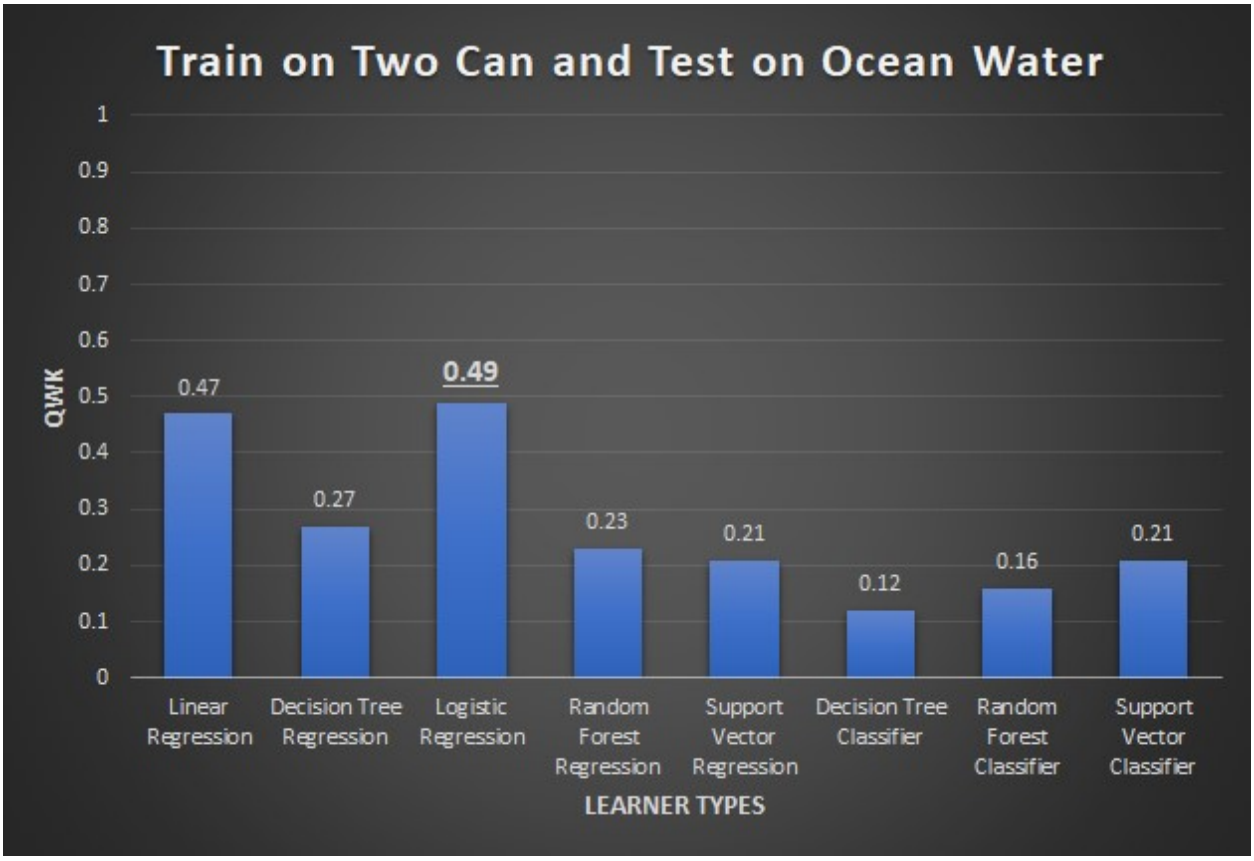


Figure 5.11: Cross domain QWK trained on Two Can dataset and tested on Ocean Water dataset

## Chapter 6: Discussion

The results of this project reveal both promising potential for industrial application and specific areas to focus on for improvement. Focusing on the positive aspects, the dataset accuracies for object classification achieved a maximum of 95.3% accuracy on the 50 images that had doubly annotated labels when using the Rules-based Classification. On the topic of feature extraction, the novel method to determine Arrow rotation was indirectly validated by the high QWK scores averaging above .7 for the 174 Image Ocean Water Dataset and peaking at .56 for the 144 Image dataset. The .56 score is near the peak .62 QWK score achieved by ETS on the 148 Image Ocean Water dataset and this score is surpassed on the 174 image dataset [38]. Furthermore, this method automatically generates the classifications and features of the objects, whereas previous versions relied on using specific software that labeled the objects and their features upon creation. Based on these two aspects alone, the goal of automating the scoring of scientific visual models is accomplished.

Although the results are promising, room for improvement remains in key areas. An essential resource for future improvement will be high-quality templates and more properly labeled ground truth visual models. Due to the unforeseeable events related to the CoVID-19 pandemic, the number of labeled visual models were limited and templates of all objects could not be provided. Based on the 20% drop in F1 Score for Squares when comparing the 50 image Ocean Water dataset to the 144 image version, it is reasonable to conclude that our perceived errors in the ground truths are valid. The large decrease in accuracy for the larger dataset cannot be properly

analyzed until correctly labeled visual models are provided, although it is likely that the accuracy will significantly increase once these are acquired. Aside from properly labeled models, high-quality templates of each object can improve methods such as Hu Moment Classification and Area Ratio Classification, both of which ultimately feed into the Cascaded Voting System. Even without these resources, the Cascaded Voting System performed relatively well compared to the Rules-based Classification, but further evaluation is necessary with better resources to determine if the Voting System outperforms the individual methods. The motivation for improving the Cascaded Voting System is due to accuracies increasing for the Area Ratios and the Voting System as the number of votes increased. This provides evidence as a proof of concept and motivates future research.

Another method that will be improved by properly labeled models will be supervised machine learning. For this project, the Decision Tree Classifier was trained only on the very few extracted, low-quality templates. These not only lacked the Diamonds shape, but were few in number. Even with these circumstances, the Decision Tree results showed a Dataset Accuracy of 94.3%. These results are promising enough to warrant future research with better resources and other classifiers, especially with ground truths containing the correct classifications and coordinates of all the shapes. Furthermore, if the individual classifiers prove effective, they could be used as inputs to the Cascaded Voting System for potential better outcomes.

A key aspect that can be explored for future improvement without additional resources is color segmentation. The average accuracy per image was lower across all methods implying that certain images are outliers. Upon further analysis, it was found that a select few images were proudcing low accuracies and they all shared the pattern of the objects being in a non-white background. Our belief is that during the binarization of the image, the objects were mixed in with

the background, thus making them undetectable. Future work can explore methods to remedy this so the background color has little to no impact on object detection and classification.

For the automatic scoring of the visual models, the two best methods were tested on the Ocean Water and Two Can datasets. Peak scores of .77 for the Rules-based Classification and .75 for the Voting System were achieved. As noted earlier, these results surpass previous methods and do so based on automatic object classification and feature extraction. These scores markedly dropped to a peak of .52 when tested on the Two Can Dataset though. The rationalization of this drop in performance is that the Two Can models have hand-drawn objects and words. This project was not designed to recognize hand-drawn objects, rather focusing on digital objects. Future work can add methods that are better suited for recognition of a broader scope of objects. If a large enough training corpus is provided, deep learning methods could also be implemented to generalize the foundational level of object classification and feature extraction, thereby generalize the automatic grading of scientific visual models.

## Chapter 7: Conclusion

The modern emphasis on scientific education is being met with novel methods of assessment such as visual modeling. Scientific visual modeling has been shown to adequately measure a student's understanding of a concept, but has been historically barred from widespread use because the models had to be graded by hand. To address this scalability problem, this project implemented a variety of automatic unsupervised and supervised methods which first classify and extract features from the objects in a scientific model. This was done by using individual approaches as well as designing a Cascaded Voting System that integrated multiple approaches in order to design a more reliable system. At the feature extraction level, this project implemented a novel method to determine the orientation of Arrows, a feature of particular importance for the final LP score prediction. Following this, the features were aggregated to better describe the overall model. Finally, a Learning Progression score from 1 (poor conceptual understanding) to 4 (strong conceptual understanding) is produced to evaluate each model.

Our results and extensive analysis are very promising and indicate the feasibility and high accuracy of our system, especially when using the Rules-based Classification and the Cascaded Voting System. For instance, using a Rules-based Classification, this project achieved 95.3% accuracy in automatically classifying objects and a Quadratic Weighted Kappa score of .77, which is higher than previously implemented methods published in literature. These results provide evidence that this method of automatically describing a scientific visual model and grading it can solve the scaling issue, making scientific visual modeling a viable tool for widespread use.

## References


1. Alajlan, N., Rube, I.E., Kamel, M.S., and Freeman, G. Shape retrieval using triangle area representation and dynamic space warping. *Pattern Recognition., 40* (2007), pp. 1911-1920.

2. Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 33, no. 5 (2011), pp. 898-916.

3. Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. From contours to regions: an empirical evaluation. *Proceedings of International Conference on Computing Vision and Pattern Recognition (CVPR)* (2009), pp. 2294–2301.

4. Augustine, N. R. Rising above the gathering storm: Energizing and employing America for a brighter economic future. *Committee on Prospering in the Global Economy of the 21st Century. Washington, DC: National Academies Press* (2005).

5. Ballard, D.H. and Sabbah, D. Detecting object orientation from surface normal. *Proc. Int. Pattern Recognition Conf.* (1981), pp. 63-67.

6. Basu, M. Gaussian-based edge-detection methods—a survey. I*EEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 32, no. 3* (2002), pp. 252-260.

7. Bay, H., Tuytelaars, T., and Van Gool, L. Speeded-up robust features (SURF). *Computer vision and image understanding, vol.110, No.3* (2008) pp. 346-359.

8. Bei, C.D. and Gray, R.M. An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization. *IEEE Trans. Comm., vol. 33, no. 10* (1985), pp. 1132-1133.

9. Belongie, S., Malik, J., and Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell., 24 (4)* (2002), pp. 509-522.

10. Biswas, S.,Aggarwal, G., and Chellappa, R. Efficient indexing for articulation invariant shape matching and retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA* (2007), pp. 1-8.

11. Bremner, D., Demaine, E., Erickson, J., Iacono, J., Langerman, S., Morin, P., and Toussaint, G. Output sensitive algorithms for computing nearest neighbor decision boundaries. *Discrete and Computational Geometry* (2005), pp. 593–604.

12. Burgers, C. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery, vol. 2, no. 2* (1998), pp.121–167.

13. Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6* (1986), pp. 679-698.

14. Cheong, C. and Han, T.D. Improved simple boundary following algorithm. *J. Korea Inf. Sci. Soc. Softw. Appl. Vol 33* (2006), pp. 427–439.

15. Cheong, C., Seo, J., and Han, T.D. Advanced contour tracing algorithms based on analysis of tracing conditions. *Proceedings of the 33rd KISS Fall Conference, Seoul, Korea, 20–21, Vol. 33* (2006), pp. 431–436.

16. Corcoran, T., Mosher, F., and Rogat, A. Learning progressions in science: An evidence-based approach to reform (2009).

17. Cover, T. and Hart, P. Nearest-neighbor pattern classification. *Information Theory, IEEE Transactions* (1967), pp. 21-27.

18. Deng, Q., Luo, Y. Edge-based method for detecting salient objects. *Optical Engineering 50(5)* (2011), pp. 301–301.

19. Deng, Y. and Manjunath, B.S. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 8* (2001), pp. 800-810.

20. Douglas, D. and Peucker, T. Algorithms for the reduction of the number of points required to represent a line or its character. *The American Cartographer, 10(42):112--123,* (1973).

21. Duda, R. O., and Hart, P. E. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, Vol. 15 (1972), pp. 11–15.

22. Fan, J., Yau, D. K. Y., Elmagarmid , A. K., and Aref, W. G. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *IEEE Trans. Image Process., 10* (2001), pp. 1454-1466.

23. Flusser, J., Suk, T. Pattern recognition by affine moment invariants. *Pattern Recognition* (1993), pp. 167-174.

24. Flusser, J., Zitova, B., and Suk, T. Moments and Moment Invariants in Pattern Recognition. *Wiley Publishing* (2009).

25. Forbus, K., Usher, J., Lovett, A., Lockwood, K., and Wetzel, J. CogSketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science 3, 4* (2011), pp. 648–666.
26. Gedraite, Estevao and Hadad, M.. Investigation on the effect of a Gaussian Blur in image filtering and segmentation (2011), pp. 393-396.

27. Girshick, R. Fast R-CNN. *ICCV* (2015).

28. Girshick, R., J. Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR* (2014).

29. Gopalan, B R., Turaga, P., and Chellappa, R. Articulation-invariant representation of non-planer shapes. *European Conference on Computer Vision* (2010).

30. Goshtasby, A. 2-D and 3-D Image Registration for Medical, Remote Sensing and Industrial Applications. *Wiley* (2005).

31. Haralick, R.M., and Shapiro, L.G. Survey: Image segmentation techniques. *Comput. Vis. Graph. Image Process*., vol. 29, (1985) pp. 100–132.

32. Hu, M.K. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions*, vol. 8 (1962) pp. 179-187

33. Jindal, S. Image Segmentation Using Improved JSEG with Fuzzy Weighted Moving K-Means (2017).

34. Karami, E., Prasad, S., and Shehata, M. Image Matching Using SIFT, SURF, BRIEF, and ORB: Performance Comparison for Distorted Images. *Proceedings of the 2015 Newfoundland Electrical and Computer Engineering Conference, St. John's, Canada, November* (2015).

35. Kim, H. Y., Araújo, S. A. Grayscale Template-Matching Invariant to Rotation, Scale, Translation, Brightness and Contrast. *LNCS*, vol. 4872. (2007), pp. 100–113.

36. Kim, J., Kim, B.S., and Savarese, S. Comparing image classification methods: K-nearest-neighbor and support-vector-machines. *Proc. of the 6th WSEAS Int. Conf. on Computer Engineering and Applications, and Proc. of the 2012 American Conf. on Applied Mathematics* (2012), pp. 133–138.

37. Klassen, E., Srivastava, A., Mio, W. Analysis of planar shapes using geodesic paths on shape spaces. *IEEE Trans. Pattern Anal. Mach. Intell., 26 (3)* (2004), pp. 372-383.

38. Leong, C. W., Liu, L., Chen, L., and Ubale, R. Toward Large-Scale Automated Scoring of Scientific Visual Models. *Proceedings of the Fifth Annual ACM Conference on Learning at Scale* (2018).

39. Lim, Y. W., and Lee, S. U. On the color image segmentation algorithm based on the thresholding and the fuzzy C-means technique. *Pattern Recognit*., vol. 23, no. 9, (1990), pp. 935–952.

40. Lindeberg, T. Edge Detection and Ridge Detection with Automatic Scale Selection. *Int'l J. Computer Vision*, vol. 30 (1998), pp. 117-156.

41. Ling, H. and Jacobs, D. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Mach. Intell., 29 (2)* (2007), pp. 286-299.

42. Littmann, E. and Ritter, H. Adaptive color segmentation-a comparison of neural and statistical methods. *IEEE Transactions on Neural Networks, vol. 8, no. 1* (1997), pp. 175-185.

43. Lowe, D.G. Object recognition from local scale-invariant features. *IEEE Int. Conf. Computer Vision*, vol. 2 (1999), pp. 1150-1157.

44. Maire, M., Arbelaez, P., Fowlkes, C., and Malik, J. Using contours to detect and localize junctions in natural images. *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)* (2008), pp. 1–8.

45. Marr, D. C. and Hildreth, E. Theory of edge detection. *Proceedings R. Soc. Lond. B. 207* (1980), pp. 187-217.

46. Martin, D., Fowlkes, C., and Malik, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 5* (2004), pp. 530-549.

47. Marvaniya, S., Gupta, R., and Mittal, A. Adaptive locally affine-invariant shape matching. *Machine Vision and Applications, 29(4)* (2018), pp. 553–572.

48. McIlhagga, W. The canny edge detector revisited. *International Journal of Computer Vision* (2011).

49. McNeill, G. and Vijayakumar, S. Hierarchical procrustes matching for shape retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1* (2006), pp. 885-894.

50. Mercimek, M., Gulez, K., and Mumcu, T.V. Real object recognition using moment invariants. *Acad. Proc. Eng. Sci*., 30 (6) (2005), pp. 765-775.

51. Merritt, J.D. Tracking students' understanding of the particle nature of matter. *Ph.D. Dissertation. University of Michigan*, (2010).

52. Mistry, D. and Banerjee, A. Comparison of feature detection and matching approaches: SIFT and SURF. *Global Res. Dev. J. Eng.* (2017), 2, 1–7.

53. National Research Council and others. Developing assessments for the next generation science standards. *National Academies Press (*2014).

54. National Research Council and others. Next generation science standards: For states, by states. (2013).

55. Oskoei, M.A., and Hu, H. A Survey on Edge Detection Methods. *Technical Report: CES-506, University of Essex* (2010).

56. Ouyang, W., Tombari, F., Mattoccia, S., Di Stefano, L., and Cham, W. Performance evaluation of full search equivalent pattern matching algorithms. *PAMI, (99):1–1* (2012).

57. Pal, N., and Pal, S. A review on image segmentation techniques. *Pattern Recognit*., vol. 26, (1993), pp. 1277–1294.

58. Palmer, P.L, Dabis, H., and Kittler, J. A performance measure for boundary detection algorithms. *Comput. Vis. Image Understand., vol. 63* (1996), pp. 476-494.

59. Papari, G. and Petkov, N. Edge and line oriented contour detection: state of the art. *Image and Vision Computing* (2011).

60. Perona, P. and Malik, J. Detecting and localizing edges composed of steps, peaks and roofs. *ICCV* (1990).

61. Prewitt, J. M. S. Object enhancement and extraction. *Picture Processing and Psychopictorics, B. Lipkin and A. Rosenfeld. Eds. Academic Press, New York* (1970).

62. Raftopoulos, A. and Kollias, D. The global–local transformation for noise resistant shape representation. *Comput. Vis. Image Underst., 115 (8)* (2011), pp. 1170-1186.

63. Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *NIPS* (2015).

64. Roberts, L. G. Machine perception of three-dimensional solids. *Optical and Electro-optical Information Processing, J. T. Tippet (Ed.), MIT Press* (1965), pp. 159-197.

65. Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. ORB: An efficient alternative to SIFT or SURF. *IEEE International Conference on Computer Vision* (2011).

66. Sebastian, T.B., Klein, P.N., and Kimia, B.B. On aligning curves. *IEEE Trans.Pattern Anal. Mach. Intell., 25 (1)* (2003), pp. 116-125.

67. Seo, J., Chae, S., Shim, J., Kim, D., Cheong, C., and Han, T. D. Fast contour-tracing algorithm based on a pixel-following method for image sensors. *Sensors, 16(3), 353* (2016).

68. Shapiro, L.G., and Stockman, G.C.. Computer Vision. *Prentice Hall* (2001), pg. 167.

69. Siddharth, M., Daniel, C., and Byung-Woo, H. Integral invariants for shape matching. *IEEE Trans. Pattern Anal. Mach. Intell., 28 (10)* (2006), pp. 1602-1618.

70. Silva, Rodrigo D. C., and George, A. P. Thé. Moment Invariant Based Classification of Objects from Low-Resolution Industrial Sensor Images. *Anais Do 11. Congresso Brasileiro De Inteligência Computacional* (2016).

71. Skarbek, W. and Koschan, A. Colour image segmentation – a survey. *Technical Report, Tech. Univ. of Berlin* (1994).

72. Sobel, M.E. Asymptotic confidence intervals for indirect effects in structural equations models. *S. Leinhart (Ed.), Sociological methodology* (1982), pp. 290–312.

73. Souza, G.B.D. and Marana, A.N. HTS and HTSn: New shape descriptors based on hough transform statistics. *Comput. Vis. Image Underst., 127* (2014), pp. 43-56.

74. Suzuki, S. and Abe, K.Topological Structural Analysis of Digitized Binary Images by Border Following. *CVGIP 30 1* (1985), pp 32-46.

75. Tang, F., Crabb, R., and Tao, H. Representing Images Using Nonorthogonal Haar-Like Bases. *IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 29, no. 12,* (2007), pp. 2120-2134.

76. Teh, C. H., and Chin, R. T. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 10 (1988), pp. 496–513.

77. Torres-Mendez, L.A., Ruiz-Suarez, J.C., Sucar, L.E., and Gomez, G. Translation, Rotation and Scale-Invariant Object Recognition. *IEEE trans.on sytems, man and Cybernetics*, Vol.30, No.1 (2000).

78. Xu, H., Yang, J., and Yuan, J. Invariant multi-scale shape descriptor for object matching and recognition. *IEEE International Conference on Image Processing (ICIP)* (2016).

79. Madnani, N., and Loukina, A. RSMTool: collection of tools building and evaluating automated scoring models. Journal of Open Source Software, 1(3) 33 (2016).

80. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J. Scikit-learn: Machine learning in Python. Journal of machine learning research (2011), pp.2825-2830.