A Comparison of Decision Trees for Ingredient Classification

Robin Bakker

June 24th, 2016

A Comparison of Decision Trees for Ingredient Classification

Robin Bakker
10548017

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 24th, 2016

Abstract

If robots were to, someday, perform duties in the kitchen, such as cooking, they should be able to distinguish different ingredients. In this thesis two decision trees for classification of fruits and vegetables are compared. The decision trees differ in their splitting structure in order to discover which type of structure is the most effective. Though the trees used identical data, the difference in structure caused entirely different classifications to be made. Classification accuracy was equal for both decision trees on the final test. Nevertheless, a tree with non-binary splits based on the object's features(color and shape) was found to be most effective, as it performed much faster and exhibited more stability than its ingredient-based counterpart.

Contents

1 Introduction

In the last decade, there has been an increase in the automation of household work [Prassler and Kosuge, 2008]. Robots performing duties such as mowing the lawn or vacuuming the floor have become available to consumers. An agent automatically performing work in the kitchen could be the next advancement in this trend and would be beneficial to people with trouble cooking for themselves, for instance, when suffering from Alzheimer's disease.

A robot butler capable of kitchen work is the goal of the Humabot challenge [Cervera et al., 2015], wherein a Humabot Nao is required to fulfill three tasks: turning off a stove, making a shopping list and cooking a meal, with guidance from computer vision techniques. Somewhat similar, are the RoCKIn@Work and RoCKIn@Home challenges [Dwiputra et al., 2014] [Schneider et al., 2014]. Also looking to progress the current state of autonomous robots, the RoCKIn@Work challenge introduces tasks such as automated collection, sorting and storing of items, whereas the RoCKIn@Home challenge requires the robot to find objects, appropriately deal with visitors and learn about the home environment. Inspired by the second task of the Humabot challenge, the creation of a shopping list, this thesis will focus on the vision requirements for the distinction and classification of ingredients.

In the Humabot shopping list task, the robot is given a list of ingredients and needs to assess whether any are missing in the kitchen. Missing ingredients are listed in a so called 'shopping list', as if the robot was determining what ingredients should be bought at the store. Fruits and vegetables from the IKEA DUKTIG set, shown in Figure 1, are used as ingredients, as they form a good substitute for real fruits and vegetables. As a result, contestants of the challenge have access to the exact same object during preparation and the final tests. Computer vision techniques are required for the agent to classify the ingredients in the kitchen and complete the task. However, a multitude of approaches is available for this classification task.

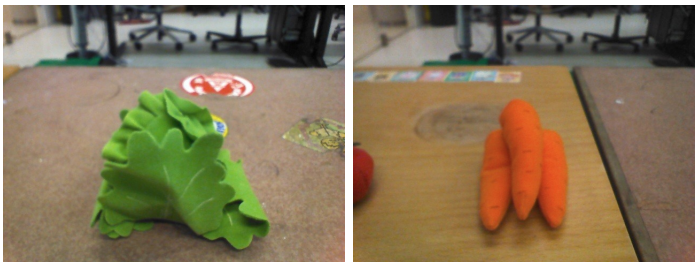


Figure 1: Example of ingredients from the Duktig set

Decision trees provide a simple, yet elegant, solution to this object classification problem. The structure of the decision tree provides an easy to follow classification algorithm for humans, which is beneficial for understanding and potential improvements or expansions of the tree. This provides contrast to methods such as neural networks that, though very effective, provide little to no insight for the user. However, when using decision trees, a variety of fea-

tures used for branching can be chosen. In the ingredient classification task, one option is to have branches based on features of the ingredients, such as color and shape. This type of tree will be further referred to as a feature-based decision tree. Another option is to use the ingredients themselves as the branching feature, essentially creating a hierarchical one-versus-all style of classification. This type of decision tree will be referred to as an ingredient-based decision tree. Figure 2 shows possible forms of the previously described decision trees.

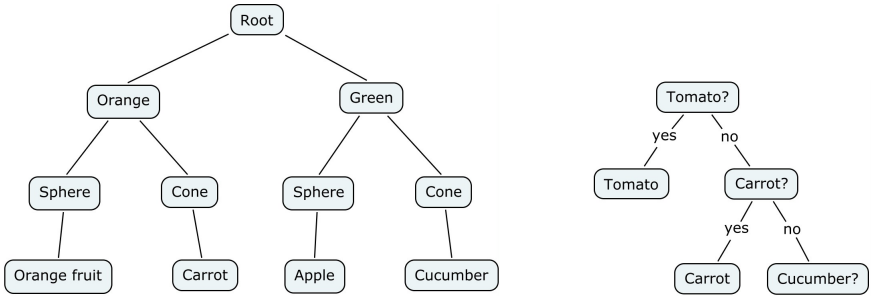


Figure 2: A feature-based decision tree versus an ingredient-based decision tree.

As of yet, no research has been performed to conclude which of these decision tree structures should be used. Therefore, this thesis will address the following research question: Which decision tree structure, feature-based or ingredient-based, allows for optimal performance during ingredient classification?

The rest of this thesis will be structured as follows: Section two introduces a theoretical background for some of the major parts of the thesis. In section three, computer vision techniques for classification problems, similar to the shopping list, are discussed. Section four gives a detailed explanation of the method. In section five a description of the performed experiments is given. The results of these experiments are presented in section six. Thereafter, a discussion of the experiment, possible further research and the final conclusion are given in section 7,8 and 9, respectively.

2 Theory

2.1 Color

Color images are commonly saved digitally as a three dimensional matrix of shape MxNx3, with M representing the image's height and N representing the image's width. Each color of a pixel at location (M,N) is represented with a vector of length 3, representing the Red, Green and Blue channels that make up the color. The values of these channels range from 0 to 255. An image of this type is referred to as an RGB image.

Color histograms, first proposed by Swain and Ballard [Swain and Ballard, 1991], can be used to describe color information of an image and is more robust than the original RGB matrix. This allows for more advanced processes

such as calculating similarity between images and object recognition. In a color histogram, bins are created to represent a part of the channel's full range. For example, a histogram with 10 bins for the Red channel of an RGB image will have one bin ranging from 0 to 24, another bin ranging from 25 to 49, and so on. Similar values are grouped together into a bin and counted, making it possible to represent an image with a set of channel vectors instead of a complete matrix, effectively reducing complexity and increasing robustness.

Though histograms of RGB images can be an effective tool for object recognition, environmental features, such as lighting, can affect performance negatively, as was discussed by Gevers and Smeulders [Gevers and Smeulders, 1999]. Alternative color descriptors have been compared for object recognition when objects faced changes in lighting intensity and color [Van De Sande et al., 2010]. As was previously shown, RGB histograms did not fare well under these changes. However, HSV histograms, representing color based on Hue-Saturation-Value channels, showed to be invariant to changes in lighting intensity. Other methods such as SIFT(Scale-invariant feature transform) [Lowe, 1999] and SIFT based methods were invariant to changes both in light intensity and color and performed significantly better on the benchmark. This is mainly due to the fact that SIFT does not rely on the color of the object and instead focuses on key-points of the object. Versions of SIFT have been integrated with color as well, though many are affected by lighting color. Only transformed color SIFT is invariant to both lighting color and intensity changes while using color information. However, this requires a substantial amount of computation compared to regular SIFT and even more compared to color histograms.

As previously mentioned, HSV histograms are invariant to changes in lighting intensity. HSV colors are represented as a cylinder, as show in Figure 3. In this cylinder, the Hue represents the base of the color and has a range of 0^o to 360^o and is calculated according to Equation 1. Saturation, calculated by Equation 2, defines the 'colorfulness' of a color. It is represented by a percentage, with 0% being white and 100% being the full base color. Lastly, value represents the brightness of the color. The calculation of the value is given in Equation 3. The value of this channel is also given as a percentage, with 0% indicating black and 100% indicating the full base color. Hue and Saturation values can be derived from RGB values directly with the formulas presented by Gevers and Smeulders [Gevers and Smeulders, 1999].

$$H(R,G,B) = arctan(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)} \tag{1}$$

$$S(R,G,B) = 1 - \frac{min(R,G,B)}{R+G+B} \tag{2}$$

$$V(R,G,B) = \frac{max(R,G,B)}{255}[1] \tag{3}$$

[1] http://www.rapidtables.com/convert/color/rgb-to-hsv.htm

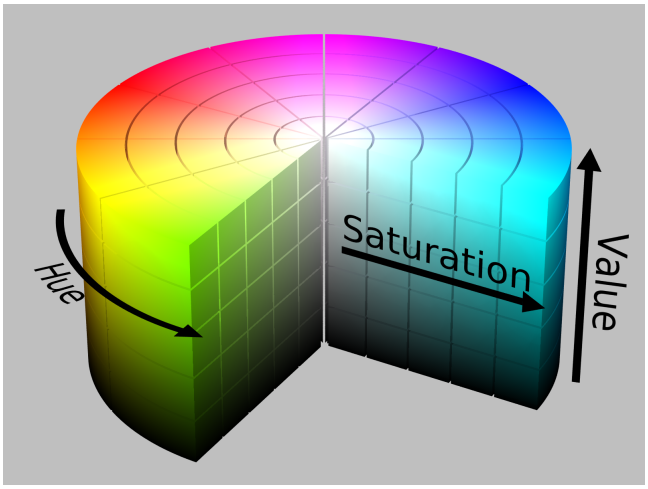


Figure 3: A representation of the HSV color scale [2]

2.2 Salience

In object recognition, important information is often accompanied by noise. Noise can be seen as all data that is irrelevant to the task of the algorithm. When actively classifying an ingredient, the agent will have to determine on its own where the ingredient is located in the image. Finding the region of interest automatically has been studied extensively and a large amount of methods were created to achieve this. In a recent benchmark, many of the state-of-the-art methods for salient region detection were compared [Borji et al., 2015]. Salience based methods, looking for objects that stand out, displayed the highest performance.

2.3 Shape

The shape of an object is often as descriptive as its color and many approaches have been taken to use the shape for object recognition [Zhang and Lu, 2004]. Both 2D and 3D images can be used for shape recognition.

Early methods often used 2D images. However, in many of the approaches, a representation of the third dimension had to be created. This creation of an additional dimension is described by Daniildis and Eklundh and requires estimations and triangulation [Daniilidis and Eklundh, 2008].

The increasing availability of low cost 3D sensors has, however, caused a shift towards three dimensional methods. In 3D approaches, three types of data are available for shape extraction and recognition: depth images, point clouds and meshes [Guo et al., 2014]. The use of depth images is the simplest as it can be taken directly from the output of a depth sensor [Stückler and Behnke, 2011]. Point-clouds, as well as object meshes, can be created from RGB-D data

[2] https://upload.wikimedia.org/wikipedia/commons/0/0d/HSV_color_solid_cylinder_alpha_lowgamma.png

collected by the sensor and can be used for more advanced processing [Schnabel et al., 2007] [Mian et al., 2010] [Johnson and Hebert, 1999].

3 Related work

Off the shelf object recognition methods were allowed for the Humabot challenge [Cervera et al., 2015]. Detection was therefore often performed by general methods such as blob detection [Lagrand et al., 2016].

However, approaches to the RoCKIn challenges have generated substantial research with regard to object recognition techniques. Shahbazian introduced a bag-of-keypoints approach to the object recognition problem of the RoCKIn@Work challenge [Shahbazian,]. Key-points were detected with the SURF detector and used in a bag of key-points approach for object classification. Negrijn used an RGB-D sensor to create and fuse color and depth images [Negrijn, 2015]. These images were then used in a template matching algorithm.

Another approach making use of key-points was proposed by Jia et al. [Jia et al., 2015]. Their algorithm was able to recognize objects from images in which the objects differed in rotation or location. This algorithm was shown to have a higher performance than many other, widely used methods.

The use of neural networks has seen success in dealing with fruits and vegetables [Brosnan and Sun, 2002]. Back propagation neural networks were able to detect defects in apples with an accuracy of 95%. Techniques using color were used to classify oranges and a method using both color and shape information was capable of predicting the quality of tomatoes. These methods were all well-defined in their own field and focused on only one ingredient. Additionally, the state of the ingredient was classified rather than its type. Another example of such a method can be seen in the paper by Pla et al., in which color and size are used to sort and grade fruits [Pla et al., 2001].

4 Method

4.1 The Ingredients

As ingredients, items from the IKEA DUKTIG fruit and vegetable sets, which contain the following items: apple, banana, carrot, cucumber, grape, kiwi fruit, leek, lettuce, mushroom, orange, tomato and watermelon, are used. From this, only the mushroom, watermelon and garlic were not used for their own reasons. The garlic and mushroom had the same color as the background, making detection too difficult for the algorithm. The watermelon had many different colors, which made classification as a single color difficult.

The ingredients, shown in Figure 4, are made of soft polyester and have an average size of 11x4x5 centimeters.

Figure 4: A display of all used ingredients

4.2 The data

Since there is no RGB-D image dataset readily available for the Ikea Duktig fruit and vegetable sets, a new dataset had to be created. The dataset consisted of a training set containing 160 examples and a test set containing 80 examples.

For creation of the images in the dataset, each ingredient was placed on a white surface with a white background. The ingredient was rotated a set amount of degrees while capturing images with the RGB-D sensor. An entry from the dataset can be seen in Figure 5

Figure 5: One of the entries in the dataset. Both an RGB and depth image are required for each example

The training set contained one RGB image and one depth image for every of 16 shots for every of the ten items, resulting in 160 examples and a total of 320 images. The ingredients were photographed starting at a 0^o offset and rotated 45^o for each image. Once a full rotation had been completed, the perspective of the ingredient was changed. In most cases, this was achieved by turning the

ingredient upside down. When the shape of the ingredient did not allow for such a transformation, another position, in which the ingredient was stable, was chosen for the ingredient.

For the test set, a rotation of 45^o was also chosen. However, ingredients in this set started at an 20^o offset in order to create novel images. Eight different shots were captured for each of the ten ingredients, resulting in a set of 80 examples and 160 images.

4.3 The hardware

Images were created using the Asus Xtion PRO LIVE RGB and Depth Sensor, which is displayed in Figure 6. This camera has an operation range between 0.8m and 3.5m, a 58^oH 45^oV 70^oD Field of View and is capable of creating both RGB and Depth video streams. Frames from the video streams can be grabbed and saved as .jpg files. The dimensions of these .jpg images are 320x240.

Figure 6: The asus xtion pro RGB-D sensor [3]

4.4 Software

The code in this thesis was written in python, version 2.7.9. Python was chosen for its flexibility and abundance of available packages for image processing. Moreover, python is one of the easiest languages to quickly start a new project in. Below, some of the frequently used packages will be introduced shortly.

OpenCV-Python
OpenCV-Python is a python package for image processing. Frequently used features from this package include image loading, image saving and the creation of histograms.

Numpy
This package enables the use of matrices in python. Matrices are extremely

[3] https://ic.tweakimg.net/ext/i/1363338345.jpeg

useful in both image processing and machine learning and their use can speed up algorithms significantly.

Openni2
The Openni2 bindings for python allowed for easy access to the functions of the Asus Xtion, such as creation of a video stream and capturing of video frames.

4.5 Approach

For the research question to be answered, the two types of decision trees, feature-based and ingredient-based, had to be implemented. The first learned to recognize the colors and shapes of the ingredients, which could then be used to build its tree structure. Ingredients could then be classified by determining the color and shape of the ingredient, which would automatically indicate which ingredient was being seen. The latter learned to recognize colors and shapes, which could be used in its ingredient comparison algorithm. Ingredient scores produced by the ingredient comparisons were used to learn thresholds for each of the ingredients. These threshold were used to create the ingredient-based tree structure. Classification in this tree was performed by calculating the ingredient score and comparing it to one of the ingredients thresholds. If the score was lower than the threshold, the ingredient was similar enough and was classified as the type of ingredient it was compared to. If the score was higher than the threshold, the ingredient would be compared to the next ingredient instead. A comparison of the decision tree structures can be made by running both algorithms on the test set. The accuracy of their classifications, as well as the time needed to classify an ingredient will provide insight into which decision tree structure should be used.

4.6 Features

As stated in the previous section, both decision trees make use of features representing the color and shape of an ingredient. In this section, the extraction of these features is explained in more detail.

4.6.1 Color

The first feature to be extracted by the algorithm is color information. Colors in the decision trees are represented in a simple fashion, making use of only basic colors. During annotation, the color of an ingredient was chosen according to the following list: green, purple, red, blue, yellow, orange, white and black. However, no white, black or blue ingredients were available in the dataset. More specific colors such as light blue or turquoise were left out to avoid further branching in the decision trees, as overlap in color was needed for the other features to have any effect. As was previously mentioned, histograms can be a successful descriptor for color information in object recognition. However, using the RGB image obtained by the camera as input for the histogram poses some problems.

First of all, as discussed by Van De Sande et al.,a regular RGB histogram provides no invariance to lighting conditions [Van De Sande et al., 2010]. A solu-

tion to this problem is a conversion from RGB to Hue-Saturation-Value(HSV). In a HSV histogram, invariance to light intensity is achieved. Unfortunately, no invariance to lighting color was implemented.

A second problem encountered when using the raw RGB image -or HSV for that matter- is the low object-to-background ratio. Since the ingredients are relatively small and the camera has a minimum distance requirement of almost one meter, a large portion of the image consists of background. This background should be ignored when learning ingredient features and classifying ingredients. Object salience in an image provides a useful solution to this problem. Hierarchical Salience Detection [Yan et al., 2013], one of the best scores algorithms in the benchmark of Borji et al. [Borji et al., 2015], is used to create a salience image from the original RGB image. A salience image, as shown in Figure 7, displays salience or 'how much something stands out' of regions in the original image. This image is in grayscale with a high intensity representing high salience.

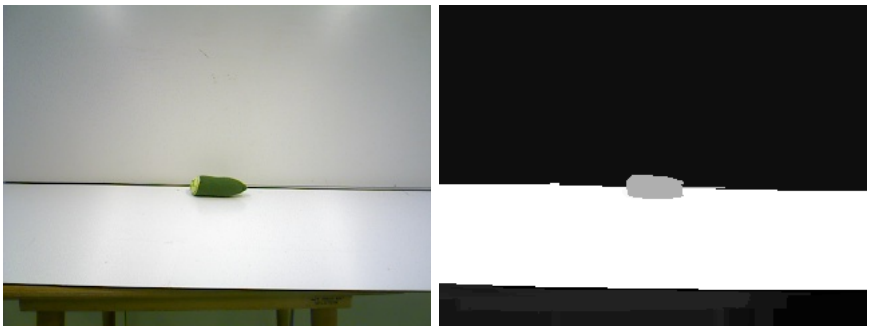


Figure 7: An RGB image and its corresponding salience image

A filter is constructed for the original image for every value in the salience image, resulting in several mappings of regions in the color image. By keeping the original colors of the filtered region and replacing the rest of the image with black, a representation of the object without the background can be constructed. From this point onward, a variety of mappings of the original image has to be used by the algorithm, as it cannot be determined automatically which mapping contains the ingredient. However, the example of Figure 7 shows that, even though the table receives the highest salience, the cucumber is recognized as a single object.

After application of the filters on the color image, two images are left for use by the algorithm, as showed in Figure 8.

Figure 8: Mapping made by the salience approach

As a color descriptor, a HSV histogram is created for each of these mappings to be used by the decision trees.

Learning color

As multiple ingredients can share the same color, histograms representing identical colors should be combined. A color is learned from an annotated training set in which salience mappings, provided by the algorithm, are given a color, as well as the object displayed in the filtered image. If the ingredient is not displayed in the mapping or more than 20% of the mapping consists of background, the mapping receives the color and ingredient label "bg" for background.

HSV histograms are calculated for all mappings. Histograms with identical annotations are combined into final histograms representing the different colors of the ingredients. This merging of histograms is done by addition of the histogram matrices, after which the resulting matrix is normalized. Lastly, the color/ingredient combinations are stored, so the algorithm can learn the features belonging to an ingredient.

4.6.2 Shape

As can be seen in the theory and related work, a variety of methods are available to describe object shapes. Depth images from the Asus Xtion, shown in Figure 9, are not able to provide enough information about the shape of an object on their own. However, the difference of normal vectors method by Boubou, et al. provides a simple algorithm that extracts important shape descriptors from the depth image [Boubou et al., 2016].

Figure 9: An RGB image and its corresponding depth image

First, normal vectors are created for each pixel in the depth image. Equation 4 shows the calculation of a normal vector at pixel p.

$$n = \begin{bmatrix} \frac{\delta d(u,v)}{\delta du} \\ \frac{\delta d(u,v)}{\delta dv} \\ -1 \end{bmatrix} \tag{4}$$

In this equation, $\frac{\delta d(u,v)}{\delta du}$ and $\frac{\delta d(u,v)}{\delta dv}$ are given by the following equations:

$$\frac{\delta d(u,v)}{\delta du} = \frac{1}{2}(d(u+1,v) - d(u-1,v)) \tag{5}$$

$$\frac{\delta d(u,v)}{\delta dv} = \frac{1}{2}(d(u,v+1) - d(u,v-1)) \tag{6}$$

In the next step, differential angles of the normal vectors are calculated for every pixel as follows:

$$d_\alpha = 1 + cos(\alpha) = 1 + \frac{n_1 \cdot n_2}{|n_1||n_2|} \tag{7}$$

$$d_\beta = 1 + cos(\beta) = 1 + \frac{n_3 \cdot n_4}{|n_3||n_4|} \tag{8}$$

with $n_1 ... n_4$ denoting the neighboring pixels of pixel p.
Finally, d_α and d_β are added to calculate the differential value at pixel p:

$$d_\alpha, \beta = d_\alpha + d_\beta \tag{9}$$

Doing this for every pixel of the depth images will result in a matrix of differential values which can be used to create a one dimensional histogram of differential values. This histogram functions as a descriptor of the ingredient's shape, much like the HSV histogram functions as a descriptor of the ingredient's color.

Learning shape

The shape in a depth image was annotated by hand and was chosen from the following list: sphere, cone, cylinder, cube and pyramid. However, no ingredients in the dataset matched with the cube and pyramid.

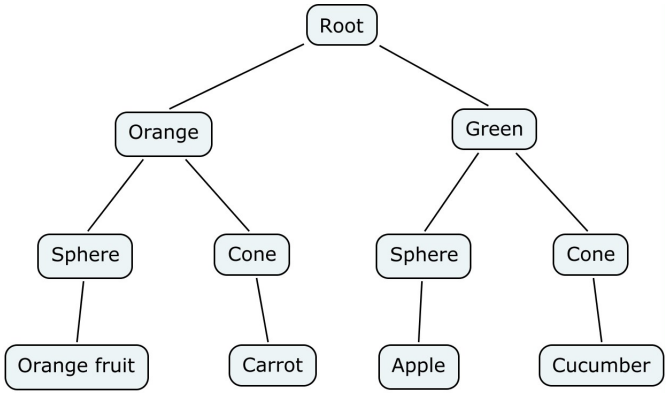
Much like with color, multiple histograms, obtained from examples, were combined into a final histogram representing a particular shape. This was done by adding histograms of shape examples and normalizing the resulting histogram.

4.7 Decision trees

In this section each of the decision trees will be explained in more detail.

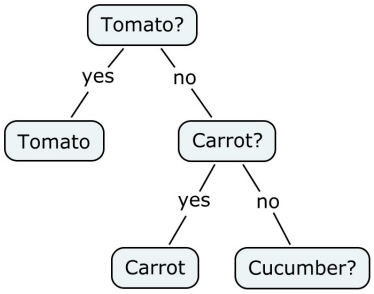
Though both decision trees make use of the same shape and color information, different approaches are taken in the use of this information.

4.7.1 Feature-based decision tree



The feature-based decision tree uses the color and shape descriptors almost directly. Every ingredient in the dataset has a unique combination of features, making it possible for the decision tree to classify an ingredient by following feature branches until a leaf node is reached. The first feature to be considered is the color feature. The color histograms of the ingredient that is to be classified are extracted, after which they are used to calculate the euclidean distance between the histogram and all learned color feature histograms. From these distances, the smallest distance is chosen, as it represents the highest similarity between the ingredient's color and the learned color. The learned color associated with this distance is returned as the classified color of the ingredient. This concludes the color choice in the decision tree, after which only the shape choice remains. The procedure for shape is equal to that of color, with the only exception that only one shape histogram is produced by the ingredient. After this histogram has been compared to the learned shape histograms and the shape belonging to the smallest distance has been chosen, the ingredient is classified as the shape node's leaf node, being the ingredient with a conjunction of the chosen color and shape as its features. This concludes the classification by the feature-based decision tree.

4.7.2 Ingredient-based decision tree



The ingredient-based decision tree makes use of the same features as the feature-based decision tree. However, instead of having branches based on these features, branches in the tree are based on the ingredients in the set. This creates a hierarchical one-vs-all classification method, in which the 'to be classified' ingredient is compared to one of the learned ingredients. When the ingredient is similar enough to the ingredient it is being compared with, the decision tree classifies it as being of that type and the algorithm ends. If, on the other hand, the similarity of the two ingredients is too little, the option is discarded and the next classification branch is taken, in which the ingredient is compared to the next ingredient in the list. This process is repeated until the ingredient has been classified or every option has been tried. In the latter case, the ingredient will be classified as 'undecided', as no classification could be made.

The calculation of similarity has, however, not been explained. Two more values are required before the decision tree is capable of classifying an ingredient: the ingredient score and ingredient thresholds.

Ingredient score

The ingredient score is a value indicating similarity between two ingredients. The score, ranging from 0 to 2, is a combination of the color distance and shape distance, meaning that a higher score indicates lower similarity between ingredients.

The first part of the ingredient score is made up of the color distance. This distance is acquired by calculating the euclidean distance between the ingredient's color histograms with the learned color histogram of the ingredient it is compared with, after which the lowest found distance is returned.

The second part of the ingredient score consists of a shape distance. The shape distance is calculated according to the same method as the color distance, with the exception that only one depth image is being compared.

The calculation of the ingredient score can be seen in Equation 10.

$$Ingredientscore = \frac{colordistance}{max\_color\_distance} + \frac{shapedistance}{max\_shape\_distance} \tag{10}$$

The color and shape distances are divided by the maximum possible distance for the respective feature, so both values are on a 0 to 1 scale. Doing this gives the ingredient score a range between 0 and 2 and ensures that both distances are evenly weighted.

During classification, the ingredient score is calculated for the 'to be classified' ingredient and the ingredient it is compared to. The score is then compared to the threshold of the ingredient to determine which branch should be taken.

Thresholds

A threshold is calculated for each ingredient in the decision tree and is used to determine whether an ingredient is similar.

Thresholds are learned from example ingredients in the training set. The ingredient score, as described in the previous paragraph, is calculated for every entry in the training set, which are then combined to form the thresholds for the ingredients. This is done for every ingredient by finding the mean of all ingredient scores obtained from examples of that ingredient. For example, in order to calculate the threshold of a tomato, the ingredient score is calculated for every tomato image in the training set. The threshold is obtained by finding the mean of these ingredient scores.

Once the thresholds are calculated, the decision tree is ready to start classifying ingredients. The ingredient's score is calculated and compared to the threshold of the ingredient it is compared with. If the score is lower than the threshold, the left branch is taken and the ingredient is classified as the type it was tested for. If the score is higher than the threshold, the right branch is taken, leading to the next ingredient comparison.

5 Experiment

After training the decision trees on the training set, experiments were performed using both the training and test set. Experiments on the training set would show the maximum capability of each algorithm, as the algorithms would be classifying on previously encountered images. Experiments on the test set would show the algorithm's performance on unseen images, as would be the case in the real life application of the algorithm.

The performance of a decision tree was based on the accuracy of its classification and the speed with which the decision was made. As classification of an ingredient could either be correct or incorrect, the accuracy of the algorithm's classification is defined by Equation 11.

$$accuracy = \frac{\text{number of correct classifications}}{\text{total number of classifications}} \tag{11}$$

Though quality of the classifications should be of main concern, speed also has to be taken into account when rating algorithms. As the algorithm has to be used by an autonomous agent in real-time, algorithms with a lower run

time would be preferential. To create insight in the time a single classification would take, speed was measured as the average run time in seconds of a single ingredient classification.

6 Results & Evaluation

As mentioned in the previous paragraph, experiments were conducted on both the training set and the test set.

Training set
When an algorithm is trained and tested on an identical set, it becomes unclear how well the algorithm performs on other sets. Optimizing results on this particular set can lead to overfitting, in which the algorithm is fine-tuned too much towards high results on the training set, often with negative influence on the general predictive power of an algorithm. Though overfitting can bias the performance on the training set, accuracy of this set can provide insight in the highest capability of the algorithm. Ideally, one would strive for a 100% accuracy on the training set whilst also minimizing overfitting. Unfortunately, results of the training set tests were not as good as this ideal.

The feature-based decision tree achieved a final accuracy of 38.1% on the training set with an average run time of 0.2106 seconds.

Figure 10 displays the confusion matrix of the feature-based decision tree. This matrix compactly displays the classifications made by the algorithm, as well as which ingredients were often mistaken for another ingredient. After analysis of the produced confusion matrix, a couple of interesting points were found. First of all, many ingredients were wrongly classified as a grape. Most striking is the fact that these ingredients did not share shape nor color with the grape, with the exception of lettuce which was assigned the shape 'sphere', just like the grape. This is most likely an indication that the color purple, a quality of the grape, was not learned correctly, as purple is very different from green both in human sight and the HSV scale. Another interesting fact is the amount of carrots that were classified as oranges. Though both share the color orange, their shapes are very different -a cone vs. a sphere-. This could be an indication that the 'cone' shape was not defined well enough. However, seeing how leek was also often mistaken for apple, it could be the case that the current shape descriptor is not effective in showing the difference between long, thin objects(cylinders and cones) and round objects(spheres). Lastly, many of the ingredients had at least one 'undecided' classification, meaning that the object could not always be distinguished from the background.

		Actual ingredient									
		tomato	kiwi	grape	apple	banana	lettuce	carrot	cucumber	orange	leek
Classfication	tomato	11	0	0	0	0	0	0	0	0	0
	kiwi	0	10	0	0	0	0	0	0	0	0
	grape	0	1	9	0	0	3	2	12	0	1
	apple	1	1	1	5	1	11	0	2	2	10
	banana	0	0	0	0	12	1	0	0	0	1
	lettuce	0	0	0	0	0	0	0	0	0	0
	carrot	0	0	0	0	0	0	3	0	0	0
	cucumber	1	0	0	3	1	1	0	1	0	1
	orange	1	0	0	0	0	0	8	0	7	0
	leek	0	0	0	2	0	0	0	1	0	3
	undecided	2	4	6	6	2	0	3	0	7	0

Figure 10: Confusion matrix of the feature-based decision tree on the training set

Accuracy of the ingredient-based decision tree was slightly higher at 46.87%, with an average run time of 2.5250 seconds.

The confusion matrix produced by the the ingredient based decision tree is shown in Figure 11. Analysis of this confusion matrix indicated that the amount of correct classifications increased for many of the ingredients. However, this increase was not seen for all ingredients, as the amount of correct classifications for the orange decreased dramatically, with the carrot often being the preferred ingredient instead. Furthermore, though the feature-based decision tree sometimes mistook carrots for oranges, this did not occur in the ingredient based decision tree. It thus seems likely the latter will always choose to classify as carrot when presented with an orange object. Confusion between the grape and several other ingredients also became apparent in this decision tree, indicating this is most likely the result of the data, rather than something inherent to the algorithm.

		Actual ingredient									
		tomato	kiwi	grape	apple	banana	lettuce	carrot	cucumber	orange	leek
Classfication	tomato	11	0	0	0	0	0	0	0	0	0
	kiwi	0	10	0	0	0	0	0	0	0	0
	grape	0	1	9	0	0	2	2	8	0	1
	apple	1	2	1	10	2	6	0	3	2	5
	banana	0	0	0	0	12	0	0	0	0	0
	lettuce	0	0	0	0	0	2	0	0	0	0
	carrot	0	0	0	0	0	0	9	0	7	0
	cucumber	0	0	0	1	0	0	0	0	0	0
	orange	0	0	0	0	0	0	0	0	1	0
	leek	0	0	0	0	0	6	0	1	0	10
	undecided	3	3	6	5	2	0	4	5	6	0

Figure 11: Confusion matrix of the ingredient-based decision tree on the training set

Test set Though the training set provided some insight in the workings and mistakes of the decision trees, the question regarding which decision tree performs better can only be answered by testing on a set of unseen images. To this end, the earlier described test set was utilized.

On the test set, the feature-based decision tree surprisingly performed slightly better than on the training set with an accuracy of 40% and an average run time of 0.4101 seconds.

In Figure 12 the increase in both accuracy and run time can be explained by a considerable decrease in 'undecided' ingredients. As more ingredients followed the complete classification algorithm, more inferences had to be made, increasing the run time but also increasing the chance of a correct classification. The increase in accuracy was, however, not that substantial. On the other hand, the slight increase indicates that the use of features in a decision tree is a stable method, not suffering from overfitting on training data. Finally, results follow the same pattern as discussed for the training set experiment. For example, the grape is still confused regularly with other ingredients, further indicating the stability of the algorithm on unseen examples.

		Actual ingredient									
		tomato	kiwi	grape	apple	banana	lettuce	carrot	cucumber	orange	leek
Classfication	tomato	6	0	0	0	0	0	0	0	0	0
	kiwi	0	7	0	0	0	0	0	0	0	0
	grape	1	1	7	6	1	2	0	7	6	3
	apple	0	0	0	1	0	4	0	1	0	3
	banana	0	0	0	0	7	2	0	0	1	0
	lettuce	0	0	0	0	0	0	0	0	0	0
	carrot	0	0	0	0	0	0	1	0	0	0
	cucumber	0	0	0	1	0	0	0	0	0	0
	orange	1	0	0	0	0	0	6	0	2	0
	leek	0	0	0	0	0	0	0	0	0	1
	undecided	0	0	1	0	0	0	1	0	0	0

Figure 12: Confusion matrix of the feature-based decision tree on the test set

Lastly, the classifications by the ingredient-based decision tree on the test set were examined. The algorithm performed with an accuracy of 40% at an average run time of 3.2301 seconds.

Compared to the experiment on the training set, a dip in accuracy can be seen for the test set, though this is often the case for machine learning algorithms. Interestingly, the accuracy obtained by the ingredient-based decision tree is identical to the accuracy of its feature-based counterpart. However, the confusion matrix, shown in Figure 13, indicates that the classifications made by the algorithm differ substantially from those made by the feature-based method. Most importantly, a significant amount of ingredients was classified as 'undecided', heavily affecting the accuracy of the algorithm.

		Actual ingredient									
		tomato	kiwi	grape	apple	banana	lettuce	carrot	cucumber	orange	leek
Classfication	tomato	5	0	0	0	0	0	0	0	0	0
	kiwi	0	3	0	0	0	0	0	0	0	0
	grape	0	1	7	6	1	1	0	4	5	2
	apple	0	0	0	2	0	3	0	3	0	2
	banana	0	0	0	0	6	0	0	0	1	0
	lettuce	0	0	0	0	0	0	0	0	1	0
	carrot	1	0	0	0	0	0	7	0	2	0
	cucumber	0	0	0	0	0	0	0	0	0	0
	orange	0	0	0	0	0	0	0	0	0	0
	leek	0	0	0	0	0	4	0	0	0	2
	undecided	2	4	1	0	1	0	1	1	1	0

Figure 13: Confusion matrix of the ingredient-based decision tree on the test set

A comparison of the results from the four experiments shows both trees have individual flaws as well as some shared ones. For example, both decision trees will mistake many ingredients for grapes. The feature-based decision tree will often mistake carrots for oranges, whereas the ingredient-based decision tree will always choose carrots over oranges. Additionally, a clear distinction was found in the run times of the algorithms, with the feature-based decision being significantly faster.

7 Discussion & Future work

In these experiments, the same data was used as input for each of the decision trees. By doing this, it was ensured that differences in classification were caused by the difference in structure of the tree. However, other kind of data could have been used for the ingredient-based decision tree. Rather than learning shape and color averages, the alternative tree could have learned ingredient shapes and colors. In other words, the algorithm would not learn 'red' and link this to a tomato, but instead learn 'tomato color'. Though overlap between ingredients of the same color or shape would have been lost, this type of data might be a better fit for the ingredient-based decision tree, improving its classifications. The implementation of this alternative algorithm has been left for possible future research.

Another aspect influencing classification is the order of the items in the decision tree. For the feature-based tree, this means putting color or shape first. Using shape as the first feature was attempted shortly but led to significantly lower results. Therefore, color was chosen as the first feature in the algorithm. Unfortunately, finding the optimal order of items in the ingredient-based decision tree was more difficult. As an item was created for every ingredient, the number of possible variations of the ingredient-based decision tree was equal to

10!, being 3628800 variations. As it was impossible to test every one of these orders, the tree was ordered based on the value of the thresholds. The lowest thresholds were placed first in the tree, making the algorithm perform its classification from strict to lenient decision making. This ordered decision tree outperformed randomly chosen orders, but no it is not guaranteed to be the order leading to the best results. Finding the optimal order of such a decision tree, for instance with AdaBoost, could be the goal of future research.

Lastly, though no experiments have been conducted on the IKEA DUKTIG set with neural networks, it would be a safe assumption to say that these models would be able to outperform these decision trees, based on the general performance of neural network [Liang and Hu, 2015].

8 Conclusion

The goal of this thesis was to answer the following research question: Which decision tree structure, feature-based or ingredient-based, allows for optimal performance during ingredient classification?

As can be seen from the results, difference in decision tree structure can lead to differences in classification, even when provided with equal input. This shows finding the optimal structure for a decision tree can be an important step in creating a classification algorithm. Finally, though performance of both decision trees was identical on the test set, there is still a lot that can be said about the differences of the algorithms by analyzing their classifications.

To answer which decision tree structure is more effective for ingredient classification, multiple aspects have to be take into consideration. First of all, performance of both decision trees was equal on the test set experiment. From this, it could be concluded that structure does not matter. However, when results on the test set are compared to the results of the training set, it can be seen that performance of the feature-based decision tree remained stable, whereas performance of the ingredient-based decision tree decreased. Moreover, the feature-based decision tree performed significantly faster classification than its ingredient-based counterpart. Taking these factors into consideration, a feature-based decision tree would currently be the best choice for an autonomous agent performing kitchen duties.

References

[Borji et al., 2015] Borji, A., Cheng, M.-M., Jiang, H., and Li, J. (2015). Salient object detection: A benchmark. *Image Processing, IEEE Transactions on*, 24(12):5706–5722.

[Boubou et al., 2016] Boubou, S., Narikiyo, T., and Michihiro, K. (2016). Differential histogram of normal vectors for object recognition with depth sensors. In *Preprints of IEEE International Conference on Autonomous Robot Systems and Competitions*.

[Brosnan and Sun, 2002] Brosnan, T. and Sun, D.-W. (2002). Inspection and grading of agricultural and food products by computer vision systems—a review. *Computers and electronics in agriculture*, 36(2):193–213.

[Cervera et al., 2015] Cervera, E., Garcia, J. C., and Sanz, P. J. (2015). Toward the robot butler: The humabot challenge [competitions]. *Robotics & Automation Magazine, IEEE*, 22(2):8–17.

[Daniilidis and Eklundh, 2008] Daniilidis, K. and Eklundh, J.-O. (2008). *Springer Handbook of Robotics*, chapter 3-D Vision and Recognition, pages 543–562. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Dwiputra et al., 2014] Dwiputra, R., Berghofer, J., Ahmad, A., Awaad, I., Amigoni, F., Bischoff, R., Bonarini, A., Fontana, G., Hegger, F., Hochgeschwender, N., et al. (2014). The rockin@ work challenge. In *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pages 1–6. VDE.

[Gevers and Smeulders, 1999] Gevers, T. and Smeulders, A. W. (1999). Color-based object recognition. *Pattern recognition*, 32(3):453–464.

[Guo et al., 2014] Guo, Y., Bennamoun, M., Sohel, F., Lu, M., and Wan, J. (2014). 3d object recognition in cluttered scenes with local surface features: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(11):2270–2287.

[Jia et al., 2015] Jia, K., Chan, T.-H., Zeng, Z., Gao, S., Wang, G., Zhang, T., and Ma, Y. (2015). Roml: A robust feature correspondence approach for matching objects in a set of images. *International Journal of Computer Vision*, pages 1–25.

[Johnson and Hebert, 1999] Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449.

[Lagrand et al., 2016] Lagrand, C., van der Meer, M., and Visser, A. (2016). The roasted tomato challenge for a humanoid robot.

[Liang and Hu, 2015] Liang, M. and Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

[Mian et al., 2010] Mian, A., Bennamoun, M., and Owens, R. (2010). On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361.

[Negrijn, 2015] Negrijn, S. (2015). Rockin@ work visual servoing.

[Pla et al., 2001] Pla, F., Sanchiz, J., and Sánchez, J. S. (2001). An integral automation of industrial fruit and vegetable sorting by machine vision. In *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, volume 2, pages 541–546. IEEE.

[Prassler and Kosuge, 2008] Prassler, E. and Kosuge, K. (2008). *Springer Handbook of Robotics*, chapter Domestic Robotics, pages 1253–1281. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Schnabel et al., 2007] Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library.

[Schneider et al., 2014] Schneider, S., Hegger, F., Ahmad, A., Awaad, I., Amigoni, F., Berghofer, J., Bischoff, R., Bonarini, A., Dwiputra, R., Fontana, G., et al. (2014). The rockin@ home challenge. In *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pages 1–7. VDE.

[Shahbazian,] Shahbazian, A. Taking up the rockin@ work object recognition challenge with the bag of keypoints approach.

[Stückler and Behnke, 2011] Stückler, J. and Behnke, S. (2011). Interest point detection in depth images through scale-space surface analysis. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3568–3574. IEEE.

[Swain and Ballard, 1991] Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International journal of computer vision*, 7(1):11–32.

[Van De Sande et al., 2010] Van De Sande, K. E., Gevers, T., and Snoek, C. G. (2010). Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582–1596.

[Yan et al., 2013] Yan, Q., Xu, L., Shi, J., and Jia, J. (2013). Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162.

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