MSU CSE 803: Fall 2015
Meal Recognition Project Final Report

4 December 2015

Team Members: Vince Fasburg, Bonnie Reiff, and Josh Thomas

Problem

For this project, students were tasked with developing a software system to automatically recognize typical meals from images. In order to limit the scope of the project, the definition of "typical meals" is confined to 14 classes of images: salad, pasta, hotdog, french fry, burger, apple, banana, broccoli, pizza, egg, tomato, rice, strawberry, and cookie. The tasks performed in order to achieve this goal included manual collection and labeling of images for the training and testing image databases, identification and implementation of the detection of differentiating features among the food classes, and implementation of a classification system that uses the output of the feature detection. After training on a set of labeled images, given an unlabeled input image, the resulting system will report a class label (where the class is one of the 14 listed above). This paper will discuss the work done by the team of students on all three of the tasks described above and present the results of the developed system on the competition dataset.

Approach

This section of the paper provides information on how each of the tasks identified in the Problem section of the report was accomplished by the team of students. All of the implementation tasks discussed below were done in Matlab.

Image Databases

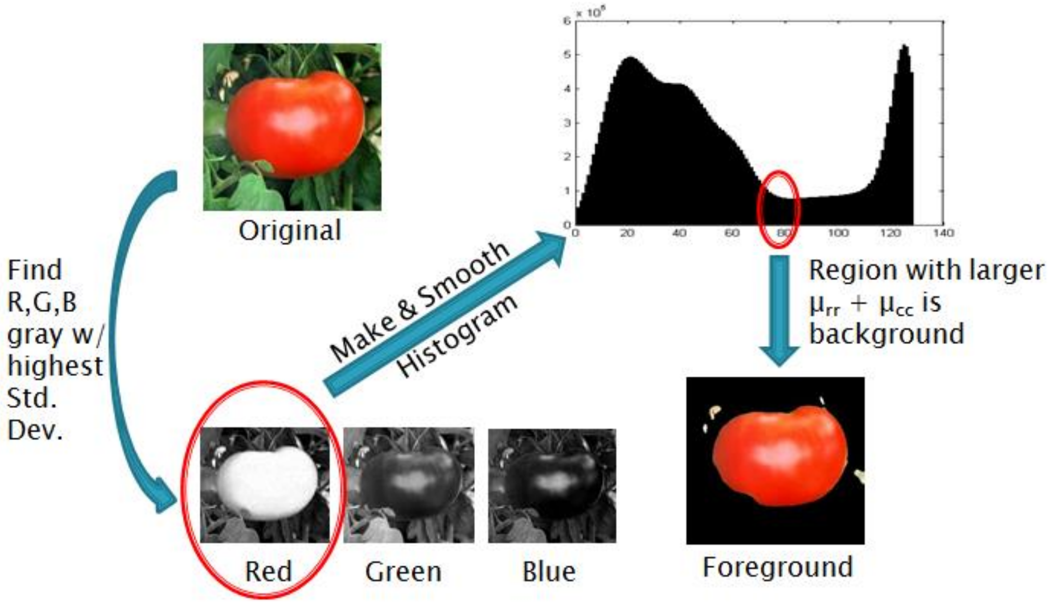
The team collected images for both the training image database and the testing image database. All images were obtained from either Google Image search or ImageNet. A minimum of 15 images were obtained for each of the 14 classes for the training database, with a total of 297 training images. While collecting images for the training database, the team primarily used images with an un-textured, monochromatic background. This allowed the system to be able to extract the features needed from the food very precisely; more complicated backgrounds have the potential to skew the actual results of a particular class of food. Each image in the training image database contains a single food item and is named according to the format "train_[class name]_[#]" (e.g. "train_tomato_2" and "train_strawberry_2"). Within the training image database, the apple class was divided into 3 different subclasses named apple red, apple green, and apple yellow. The reason for this is because of the drastic difference in color between different types of apples. This division allows the feature values for each subclass to be calculated separately and appear as a separate cluster in the data, thus minimizing the variety of the images in the apple class and promoting better classifications. Similarly, the egg class is broken into egg and eggshell classes. For the testing image database, a few more difficult types of images were allowed such more complicated backgrounds, images with shadows, and images with more in the foreground than just the food (i.e. plates and utensils). The testing dataset has a similar naming scheme to that of the training database, with a total of 82 images.

Foreground Detection

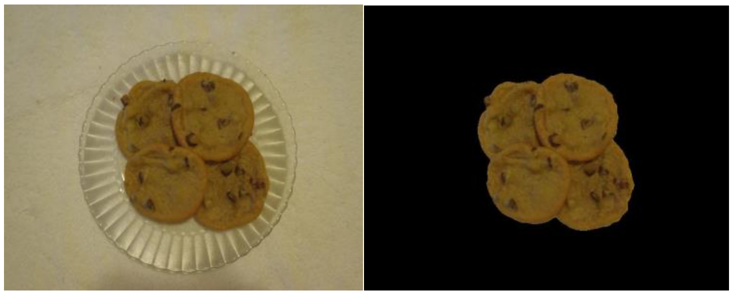
In order to recognize the foods in each image, the first task is to separate the food items from the background elements of the image. This is accomplished by first creating a grayscale image from the colored image. In order to ensure that the food stands out prominently from the background, the grayscale image is not computed as it normally would be from the RGB image. Instead, each color, red, green, and blue, is tested as a possible value to use as the grayscale image. For each color plane, the standard deviation of the pixel intensities is calculated, and the grayscale image with the highest standard deviation is chosen to be the image from which the background and foreground are computed. This relies on the estimation that the highest standard deviation corresponds to the RGB grayscale image which is most chromatically different from its background. This estimate proved to be very successful in practice.

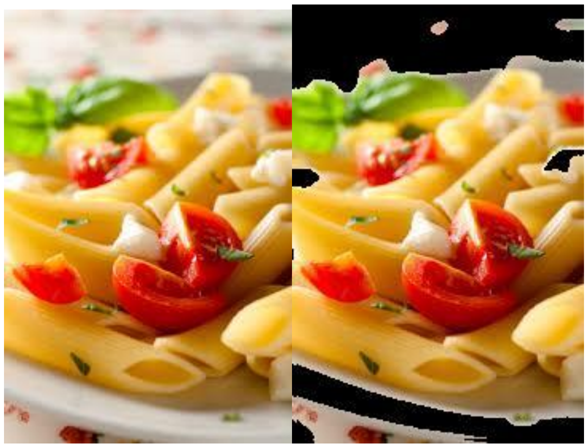
Once the grayscale image decision is made, a histogram is created from the intensities of the pixels in the image. This histogram is then smoothed several times by Gaussian filters of varying lengths so that small differences between neighboring intensity bins will be removed. The bin with the least energy is then selected as the threshold to separate foreground from background, with the region with the larger row and column moments considered to be the background since it is the less centralized of the two regions. If this lowest valued histogram bin occurs in the first or last bin, then the image is considered to have no background (such as a zoomed in image of fries or rice).

This process is shown in the figure below.

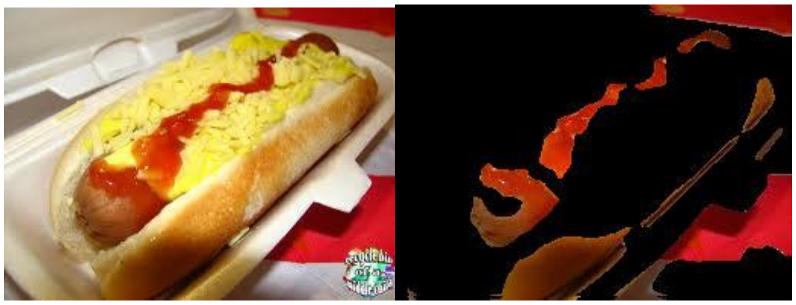

Original
Find R,G,B gray w/ highest Std. Dev.
Make & Smooth Histogram
Region with larger $\mu_{rr} + \mu_{cc}$ is background
Red
Green
Blue
Foreground

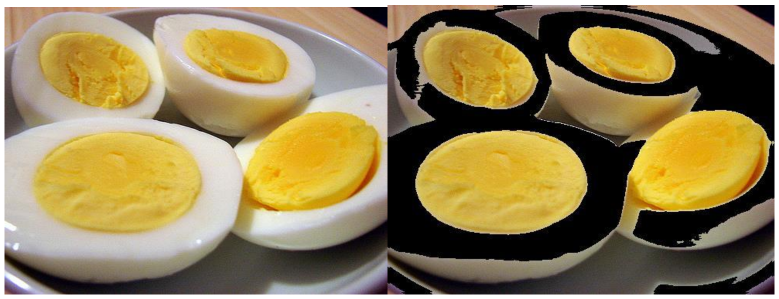
This technique proved to perform very well when the background was monochromatic, and fairly well when the background was more complicated. Below are some examples of the foreground region finding that worked well.





There are some issues with this technique. First of all, fining multiple separate foods in a single image is complicated. For example, the lowest and second lowest histogram troughs could be used to separate two foods from each other as well as from a background, but with no way of knowing in advance how many different foods are in an image, this method would not be able to make meaningful distinctions. It is also possible that this method may fail to show the foreground food at all, such as in cases where the background colors varied so much or were so similar to that of the food, that the food itself was accidentally categorized as background. There were few examples of this occurring in our data, but below are some of the worst samples. Images such as these in which the chosen foreground would not be an accurate addition to the training data were removed from the image database.





Feature Detection

The following sections describe each of the three main features used to classify the images.

Color

The first feature that is used to classify testing data is color. The approach used for this feature is very similar to the color histogram approach taken to detect faces in homework 3. The RGB color values are combined into a single 6 bit number by concatenating the 2 most significant bits of each of the red, green, and blue colors of the foreground portion of the original image. Each bin of the resulting 64-bin histogram makes up the color feature for each testing image.

This method was chosen after first trying another method, which when tested, did not end up performing as well as the method described above. This method involved computing a color histogram in a different way. For this purpose, the color for a pixel was calculated by converting the image to HSI format, and then multiplying the hue, saturation, and intensity of each pixel together. The resulting value represented the hue of each pixel, but with less weight given to a pixel which had either low intensity (being closer to black than white) or low saturation (being closer to gray than to colored). This value (hue * saturation * intensity) was added to the appropriate bin for each pixel, where the appropriate bin was determined by the hue value alone, divided equally into 60 bins between 0 and 360 degrees. The resulting histogram generally resembled one or more Gaussian distributions, of which the mean and standard deviation were computed and used as features in the overall feature vector. The team also experimented with using the values of each of the bins in the histogram as features, as is done in the final color feature described above.

Texture

In the planning stages of the project, the team identified texture as another feature that could help to differentiate between the image classes. For example, this feature should be useful between classes of similar color and shape, such as apple and tomato, where the color and shape are similar, but tomatoes are typically smoother in appearance. In addition, food classes such as strawberry, french fry, salad, broccoli, rice, and pasta should exhibit a more complicated texture than some of the other food classes.

The final system contains a very simple, but effective, measure of texture to differentiate between the various food classes. The function first applies Matlab's range filter to a grayscale version of the input image. This returns a matrix of the same size as the input image, where every cell is set to the maximum range between all pixels in the 3x3 neighborhood around the corresponding pixel in the original image.

Using input from the foreground detection to determine the number of pixels in the image that are part of the foreground region, the sum over all cells in the range filter output is then divided by the number of pixels in the food image to obtain a measure of the texture value per unit area of the foreground region.

In an effort to improve the system, Law's Texture Energy Measures method was also implemented by the team. These measures produce 9 "energy map" images corresponding to the application of filters formed from convoluting pairs of four feature detector vectors (level, edge, spot, and ripple). Once again using edgeness per pixel in the food region, a summarizing value was computed for each one of the nine energy maps, thus resulting in a 9-dimensional feature vector for each image. An analysis of the means and standard deviations of these computed values among and between image classes was done in order to find the maps with the highest interclass variation and the lowest intraclass variation. Using the top three maps from this method as features in the system, testing revealed insignificant improvement over the range filter method and thus the other method was chosen for efficiency reasons.

Shape

Finally, the team identified shape as the last feature to help classify the food images. There were two shape features that were considered; circularity and tangent angle histograms. The goal of these features was to separate classes that may have similar colors and textures such as banana, yellow apple, and french fry. The circularity function is a very simple implementation. The function takes as input the mask from the foreground detection operation, which indicates which pixels belong to the foreground and which belong to the background (as 1's and 0's). Using the erosion operator, and then subtracting the erosion image from the original image, the border of the foreground is obtained. The mean radial distance from the centroid of the foreground and the standard deviation of the mean radial distance were then calculated from these border pixels to then calculate the circularity.

For the tangent angle histogram, the same erosion operation was used to obtain the border of the foreground. A tangent angle was calculated by looping through all border pixels and calculating the tangent between that pixel and the fifth subsequent pixel ahead (skipping 4 pixels between each set of pixels). The tangent angles were then normalized in relation to the axis of most inertia (the axis of most inertia is used as zero degrees). This allowed the tangent angle histogram to be rotation invariant. These normalized tangent angle values were then grouped into 8 different bins of 45 degree width. For objects that were mostly circular, all bins should be approximately the same height. For objects that are longer, the bins around 90° and 270° should be higher than the rest.

After both shape features were implemented, it was found that the tangent angle histogram did not produce as good of results as the team had hoped. Further analysis led the team to believe that because the results obtained from the tangent angle histogram rely on a very precise background and foreground separation, small deviations from the border of the food item decrease the power of the features. In some cases, the use of the tangent angle histogram actually made the overall results worse, so in the end the team did not end up using the tangent angle histogram in the code and focused only on the use of circularity as a measure of the shape feature.

Image Classification

During the training phase of the recognition system, a 66-dimensional feature vector is computed for each image in the training dataset using the techniques for color, texture, and shape that were described in the previous sections. Based on the known labels, the means and standard deviations of the vector values are stored for each class.

In order to classify an image, a feature vector for the input image is generated using the same 66-dimensions as used for the training data. This vector is then compared to the mean vector value of each class using the Euclidean distance, and each feature distance is then scaled (divided) by the standard deviation of that dimension. The scaling by standard deviation is used to address the fact that the density of the each cluster of images in a class is not uniform.

Since the different values within the feature vector have very different scales and distributions, the classification applies additional gains to normalize the scale of the distances on each feature. For example, any given value in the normalized color histogram cannot exceed 1/64, while the values of the range and shape features can have much higher values. To mitigate this problem, each distance value is divided by the maximum value of that feature for any sample of any class. This ensures that classification is not affected by the differing sizes of each value in the frequency vector.

Once each feature in the feature vector is normalized to be on similar scales, additional weights are applied to the distance between the current sample and each class mean of the training data. These weights are manually tuned, and are used to give more weight to the values in the feature vector that are better at discriminating between classes than others, and to account for the fact that the color, shape, and texture features contribute a different number of features to the overall feature vector (color contributes 64 values, while shape and texture each contribute 1). In the final recognition system, there is a weight of 1/64 assigned to each color feature (for a total weight of 1 for all color features), a weight of ½ assigned to the texture feature, and a weight of ½ assigned to the shape feature. Once the scaling and the additional weights have been applied to the distance values, the image is classified as the class whose mean is closest to the feature vector of the current image, using this scaled method.

Results

The output values from training program are shown in the table below. The team chose to use a closed-set system, meaning that there is one entry per food class and no rejection class. Each row of the output corresponds to a single class and prints the true detection rate (“Accepted”), the true rejection rate (“Rejected”), and the average between these two numbers. The true detection and rejection rate were computed by comparing the expected class label generated by the recognition system to the actual class label provided along with the competition dataset for each image. The averages of the true detection rate and the true rejection rate are the numbers that are output to the competitionResults.txt file included in the project submission.

Class	**Accepted**	**Rejected**	**Average**
apple	40.91	83.68	62.30
banana	30.77	87.66	59.21
burger	36.36	83.26	59.81
frenchfry	33.33	83.54	58.44
pizza	60.87	81.93	71.40
pasta	-	-	-
hotdog	31.25	86.12	58.69
strawberry	60.00	88.38	74.19
tomato	41.67	85.94	63.81
egg	41.67	84.34	63.00
salad	36.36	73.22	54.79
cookie	-	-	-
rice	45.45	85.36	65.41
broccoli	69.23	86.81	78.02
Total	**43.99**	**84.19**	**64.09**

The average among all of the class detection results is 64.09, with a minimum of 54.79 (salad) and a maximum of 78.02 (broccoli). Note that the values for the cookie and pasta have dashes because the resulting recognition system does not handle these two food classes.

All training and testing operations were run on MSU's engineering computers, which have 20 available cores. The programs were written to take advantage of Matlab's parallel processing toolbox, using parallel for loops to process as many images in parallel as possible. In regards to efficiency, the training program takes 13.14 seconds to process the 297 images and create the class data statistics needed by the testing program. In order to classify the 261 images in the competition dataset, the test program runs for 13.94 seconds, where the ending time is considered to be the point at which all images have been classified, but the output has not yet been printed to the Matlab console. This results in an average of 0.053 seconds to process and classify each individual image.

Observations and Discussion

Performance of the System

Although our system performed fairly well with the collected image database, as with any system, there are several limitations and areas for improvement that have been identified by the team. One such limitation of the system is that because the features should only be evaluated for the food region as opposed to the entire image (which may contain a complicated background), the performance of all of the feature types discussed above in the report depends on the foreground detection mechanism. In cases where the foreground mechanism did not perform well, shape was the most impacted feature type (i.e. if the mechanism detects the plate that pasta is sitting on as opposed to the individual pasta shapes). Manually reviewing the image database reveals that a good shape classifier would have been a good secondary feature type to color and thus the limited ability of this feature was detrimental to the performance of the system.

The other major limiting factor in the results is the robustness of the training image database. In the team's testing and tuning of the recognition system, it became very apparent that small variations in the images used affected the number of images recognized and rejected for each class. The ideal goal in collection was to obtain a set of images dissimilar enough to prevent overfitting, but similar enough to ensure a high rate of true detection. In practice, the team found this to be harder than expected, which resulted in a significant amount of image database refinement during the period of testing the developed system on the validation/testing image database.

In terms of potential modifications to the system, the classification system has some room for improvement. As alluded to above, one limitation of the chosen classification system is that the same feature types and respective weights are used in all classification decisions. Instead, learning a decision tree-type classifier may be useful. This type of classifier uses only the features necessary to classify an image and can use different weights/threshold values between classes in different situations. For example, if the food is green, the tree may use shape as the next feature to distinguish between an apple, broccoli, and salad. But if the food is red, texture may be used to distinguish between strawberries, apples, and tomatoes.

In addition, the ability to reject images of an unknown/unlearned class can be added to the system to make it an open-set, rather than closed-set recognition system. For the scope of this project, this was not necessary due to the assumption that all images belong to one of the 14 specified classes, but would be a useful feature for the system nonetheless. Under the current classification system, a reject class would be composed of images that do not belong to any class based on the distance of the feature vector to the mean of every class of images; the threshold to determine the maximum distance from the mean of any class would need to be experimentally determined.

Lessons Learned

In the development of the meal recognition system, one of the most important lessons for all of the team members was that simple feature measures can outperform more complicated, involved ideas. As explained in the report, experimentation with various methods was done for each feature type (color, texture, and shape), with the end result generally being the use of the more obvious method. This is not to say that simple features should always be used, but that their power should not be underestimated. In addition, when attempting to determine which methods to use for each feature, it became clear that manual feature selection can be time consuming and involved.

At a system level, the team also learned the importance of testing the interactions between various portions of the system, as well as testing individual portions. For example, the team found that although a feature function may work well individually on the test set, performance may decrease with the addition of other features. The attempt to control this phenomenon led to the introduction of the feature weighting system discussed above in the Image Classification section.