Image Segmentation and Area Estimation towards a Calorie Counting Application

By: David Hackenbracht and Tanner Harvey

Introduction

Counting calories is an effective weight loss strategy, which is often complicated by the tedium of portion measurement and the time it takes to research each food's calorie content. Certain apps, like MyFitnessPal, seek to expedite this process by offering users a way to type their foods into the app, pulling up a large database from which users select a closest match and estimate their portion size. This app also allows users to scan the barcode of certain foods to find better matches. While MyFitnessPal makes the process of counting calories easier, we seek to improve this process further by allowing a user to count their calories simply by taking a picture of his or her plate. Institutions such as MADiMa (which holds an International Workshop on Multimedia Assisted Dietary Management) and others have made significant progress towards this goal, but as of yet there are no commercially available applications that estimate calorie count from an image. Since a complete implementation of such an app is complicated beyond the scope of our project, we seek to implement two features necessary to its realization: segmentation of an image of a plate, and an estimation of the area created by these segmentations.

Implementation

In this paper, "food" or "food items" refer to a distinct portion of a meal contained in its own region on a plate; by this definition, a salad is a "food," rather than its constituent parts (lettuce, tomato, etc.). A total of seven plates of food, each with three distinct food items, were analyzed. Six plates are images from the same website (used

because of the similarity of the images) and one plate we photographed.[1] All plates are well lit, photographed from above, and the foods on the plate are relatively contained in their own region with some overlap. The six plates from the website are all on a white plate with an opaque white background and are almost completely full of food, where our photograph contains a maroon plate, a white background, and a significant gap between foods. The foods are varied in color and texture, except for a few notable examples.

The image segmentation and area estimation process is summarized below in Figure 1. Complications arose during preliminary experimentation with the images due to the similarly colored plates and tables. Thus, we created the MATLAB program isolatePlate.m to create a circular mask around the region of the plate, effectively changing the color of the table in the background to black. Isolating the plate required finding its diameter, which we found to be 600 pixels with the MATLAB drawline function. Since the average diameter of a plate is recorded to be 10 inches long, we had a conversion ratio to calculate the estimated area of the food at the end of the segmentation process.[2]

The program plateSegAndArea.m is responsible for our remaining calculations and the bulk of the segmentation process. Given the masked image returned from isolatePlate.m, we perform a rudimentary k-means segmentation (via the function imsegkmeans) to see how well the image could be segmented given its color. We found color alone was inadequate to segment the food, so we endeavored to segment by texture as well. The texture-based segmentation is achieved by means of passing a

grayscale image through a Gabor filter, which looks for “frequency content in the image in specific directions in a localized region.”

The results of the Gabor filter allowed us to perform another k-means segmentation with information gathered from both color and texture; we ultimately use these results to find the isolated regions of specific foods. These isolated regions are likely to contain extraneous segments that don’t belong to the primary region we consider the food to be. Thus, we took the images with the isolated regions, converted them to binary images, dilated the images with a 9x9 neighborhood, performed connected components analysis to isolate the largest region (the area we suspect the food is in), and erode the image (again with a 9x9 neighborhood) to compensate for the earlier dilation. The resulting black and white images portray the region the food occupies (colored white), and since these images are of the logical type, the sum of all the 1’s in their respective matrices is equivalent to the number of white pixels in the image. Since the number of pixels is inherently an area, we can easily convert to square inches given the pixel to inch ratio we measured earlier.

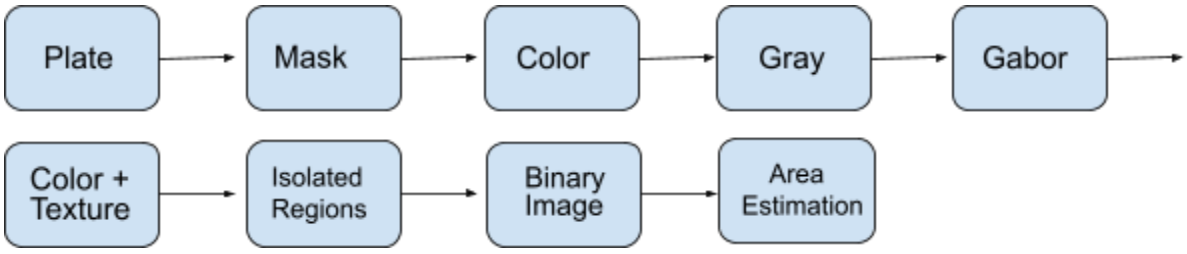

Figure 1. Steps of Food Segmentation and Area Estimation.

Parameters

The only hard-coded parameters in our code are the wavelengths and orientations used by the Gabor filter. The orientations 0, 45, 90, and 135 are reasonably distinct angles the Gabor filter can use to distinguish textures occurring at the six different wavelengths (3, 6, 12, 24, 48, and 96) which were recommended in the MATLAB documentation.[4]

The function isolatePlate.m uses two parameters: image and radiusSize. The inclusion of the image should be obvious; it seeks to isolate a circular region in an image and add a black background outside the circle. Further, radiusSize (which is the size of the radius of the circle in pixels) was included since the diameter of the plates was known to be 10 inches, and we knew 600 pixels amounted to 10 inches.

The function plateSegAndArea uses three parameters: I (the image), numItems (the number of food items on a plate, plus one for the plate, and plus one for the background; functionally numItems = number of food items + 2, and inchesToPixels (the ratio of inches to pixels). Again, the inclusion of the image should be obvious, since the program seeks to segment the food in the image into separate regions. The number of items is included to allow for variations in the number of food items on the plate, even though all the plate images we used had three items. Since numItems is passed to the imsegkmeans function (which we ideally want to segment the plate based on the number of food items) it is appropriate for this value to be an input variable. Finally, inchesToPixels is included to allow for a conversion ratio so the function can calculate the area the food occupies in terms of square inches.

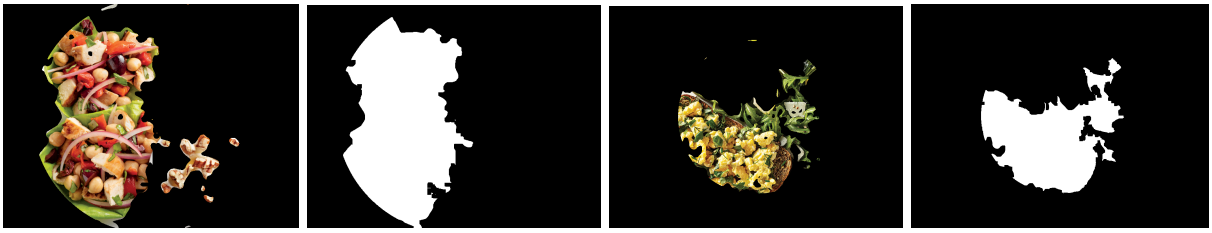

Figure 2. Area Estimation Using Binary Image Representation.

Results

The results of this project were mixed. The accuracy of the segmentation was determined by visual inspection of the images, and the accuracy of the area depended entirely on the segmentation process (see Figure 2). Some images were segmented very well, as in the case of plate3.png (see Figure 3). This image proved promising from the early stages of segmentation; the food (green beans, rice, and a pork chop) is separated in three distinct regions and has notably different textures and colors. Others, like plate5.jpg, which contains a salad, garnished chicken, and sweet potato fries, were not segmented well. The algorithm fails to distinguish the garnish on the chicken from the contents of the salad. Further, it segments parts of the chicken, the fries, and half of the salad together, and results in one segment (the third image in Figure 4) that only contains a tiny portion of the plate. In total, seven plates were photographed, each including three foods for a total of twenty-one foods. Out of twenty-one foods, ten were determined to have been correctly segmented after visual inspection.

Figure 3. Segmentation of plate3.png.

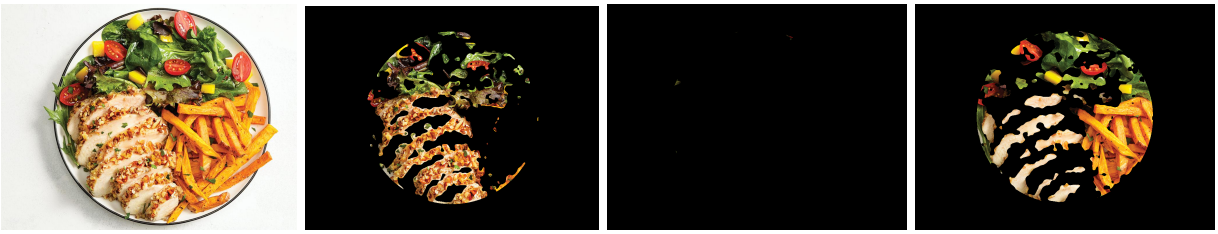

Figure 4. Segmentation of plate5.jpg.

Ideas for Future Improvement

Segmentation and area estimation are only two steps towards a fully functioning calorie counting application. This kind of app would need to implement food recognition to be effective, a feat that would likely benefit from the use of a large neural network. Such a network would undoubtedly require a significant amount of data, given the vast diversity of food. Beyond food recognition, the application would need access to a database of food nutrient information. Several of these databases already exist; of note, MyFitnessPal has such a database, which contains information for other macronutrients like fat and protein content - much more than just the calorie count. Further, while we manage to calculate area based off our images, a robust calorie counter would need to calculate the volume of the food, a much more challenging task. However, recent camera technology promises the ability to perceive depth, which would make the task of calculating volume more accessible. Finally, the ideal application's segmentation algorithm needs to have better accuracy than ours was able to achieve. Potentially, this application could prompt users to tap on the center of various food regions, or draw a perimeter around the food to better capture the correct region.

Closing

While there are many applications that help users count their calories, computer vision technology promises to help even further. Indeed, the available applications are severely limited by the ability of a user to correctly estimate his or her portions, a feat that computer vision is well suited to overcome. While the full automation of a calorie counting application has yet to be achieved, there are many scholars and companies devoting time and effort to make it a reality. It is likely only a matter of time before we will be able to scan our plates and know how much we really ate.

Citations

1. Intechnic, http://www.intechnic.com, and American Diabetes Association. “Perfect Meals by the Plate.” *Diabetes Food Hub*, 1 Aug. 2020, https://www.diabetesfoodhub.org/articles/perfect-meals-by-the-plate.html.
2. “What Size Is a Dinner Plate?” *Made In Cookware*, https://madeincookware.com/blogs/dinner-plate-size.
3. “Gabor Filter.” *Wikipedia*, Wikimedia Foundation, 5 July 2021, https://en.wikipedia.org/wiki/Gabor_filter.
4. “Imsegkmeans Documentation.” https://www.mathworks.com/help/images/ref/imsegkmeans.html.