IdentifyBox Visual System (IBVS)

Guled Ahmed

[ahgu1201@stcloudstate.edu](mailto:ahgu1201@stcloudstate.edu)

Edmund Olson

[oled1001@stcloudstate.edu](mailto:oled1001@stcloudstate.edu)

Damilola Okusanya

[oookusanya@stcloudstate.edu](mailto:oookusanya@stcloudstate.edu)

**Abstract**

Identifying an object within an image may seem like a simple computational problem, but the underlying theory and methodologies behind it are deceptively complex. An understanding of how a biological vision system works sheds light on the intricacies of constructing an artificial vision system. Object identification, as a computational problem, requires much pre-processing in order to remove as much background noise as possible. Once pre-processing is completed, a methodology for object identification is necessary. One such methodology is to use a small sub-image reflecting the object to be identified, and comparing it against each position on the original image. This method is known as template matching, which is used as the identification technique in the IdentifyBox system. Given seven images of cardboard boxes in various situations, the IdentifyBox Visual System was able to identify four cardboard boxes.

**1. Introduction**

Computer vision is the construction of explicit meaningful descriptions of physical objects from images (Szeliski, 2011). The main objective of computer vision is to build computational models of the human visual system in order to conduct particular tasks. This objective is not a trivial matter, there are many complications in creating such a computational model. One of these complications includes the fact that we, as humans, cannot introspect about vision. This increases the difficulty in creating a computation model to emulate the human visual system. This lack of introspection is followed by years of research on how the brain works in order to understand the human visual system.

Studies on the human brain include examining the brain's structure and connections by staining a small fraction of neurons in the brain, stimulating cells in the brain of patients during surgery, and correlating brain damage with cognitive changes in brain damage. These studies has elevated our understanding of how the brain works. Conversely, this allows us to understand how the human visual system works since it is a vital system of many different systems within the brain. Furthermore, the formation of an image involves an ample amount of information such as depth and noise. Moreover, many factors are involved in forming an image, such as the brightness value of a pixel. In addition to the difficulties in forming an image, the concept of mapping from 3D scenes to 2D images also presents an issue in that the mapping is of many-to-one. In an engineering standpoint, there is a lot of information to manage. Despite these complications in building a computation model of the human visual system, the main challenge of computer vision is one of explicitness (Szeliski, 2011).

Exactly what information about scenes can be extracted from an image using only very basic assumptions about physics and optics? Explicitly, what computations must be performed? Then, at what stage must domain­ dependent, prior knowledge about the world be incorporated into the understanding process (Szeliski, 2011)? All of these questions are addressed in forming a solution to identifying an object, specifically a cardboard box, within a digital image.

**2. The Problem**

The identification of objects in digital images can be approached as a computational problem. Many techniques can be considered when solving the problem of actual identification, but first, it is always necessary to pre-process the image. Pre-processing involves creating a simple image, removing as much unwanted material from it while retaining the base characteristics of the object to be identified. Pre-processing involves applying filters to the image one at a time. Helpful filters include histogram equalization, smoothing, contrast, thinning, and edge detection. Typically, the objective is to end processing with an output image that resembles a black and white line drawing.

One such identification problem is that of identifying a cardboard box. This is attempted on seven given images taken of the same cardboard box in different places and resting at different angles taken around St. Cloud State University’s campus. The different images primarily present challenges in what kind of “noise”, or unwanted marks exist on the image’s background apart from the cardboard box.

**3. Processing Steps**

In order to facilitate cardboard box detection, it is necessary to create filters which can be applied to the images. These filters are operations that can be performed to create certain effects on an output image, making them more ready to enter the template matching identification stage.

Due to the heavily varying nature of noise and shadows in each image, it was necessary to tailor the filters to each image for maximum ease of identification. Specific values were changed, primarily in the contrast filter, however the order in which filters were applied was generally similar for each image. Usually, the sequence of filters applied consisted of contrast, followed by smoothing, followed by Kirsch edge detection. In some cases, it was necessary to switch the order of application of contrast and smoothing. This was because in images that had excess noise on the cardboard box needed to be made a more uniform shade by smoothing before further operations. When tweaking values of the contrast function, a similar goal was identified. Setting “threshold” values in a contrast function was mostly dependent on the existence of “noise” that interrupted the consistency of faces or edges of the cardboard box.

**3.1 Contrast Adjustment**

In the process of bringing out certain features in an image, having a function that performs contrast can be invaluable. Increasing the contrast of an image, in essence, makes the light sections lighter and the dark sections darker. Typically, a basic contrast function will iterate over each pixel of the image, and increase its brightness value if over a certain threshold and decrease the value if below a certain threshold. The most generic example of this would be to have a threshold value of 127 (the mid-point between black, 0, and white, 255) and add or subtract an integer constant depending on where it lies in relation to the threshold. Also, it is important to have correction for values that are out of bounds. If a constant is added to a pixel’s brightness value and it goes above the maximum allowable value (255 in this case), it will be set to that maximum value. This ensures predictable output.

for (int i = 0; i < image.length; i++){  
 for (int j = 0; j < image[i].length; j++)

{  
 if(image[i][j] > 127)

image[i][j] += 10;

if(image[i][j] < 127)

image[i][j] -= 10;

//correction for out-of-bounds values

if(image[i][j] > 255)

image[i][j] = 255;

if(image[i][j] < 0)

image[i][j] = 0;

}

}  
return image;

**Figure 1:** Basic implementation of a contrast function with out-of-bounds correction.

A similar but slightly different approach is to have two separate threshold values. For example:

if(image[i][j] > 130)

image[i][j] += 10;

if(image[i][j] < 120)

image[i][j] -= 10;

**Figure 2:** Two threshold values.

This approach is usually used in special cases where there’s a segment of the image being divided along a contrast threshold line where it’s more desirable to preserve a similar brightness across the segment. Since the values from 120 to 130 are not affected, areas with values in that range won’t be “broken up” by a harsh contrast change.

**3.2 Histogram Equalization**

Histogram Equalization is a low-level image processing technique used to enhance the contrast of an image. In a grayscale image, each pixel value represents the intensity of that particular pixel. The range of pixel values in a grayscale image starts from 0, being the darkest value, and 255, being the lightest value. Pixel values that are low are considered to have lower intensities than those that are higher. The histogram of an image can be represented as a graph where our pixel intensities (values) are the independent variables and the frequencies of these intensities are the dependent variable. The implementation of the technique goes as follows:

Firstly, a histogram is built using an array that is the size of the range of pixel values available (256 (zero based)). This is followed by computing the histogram by scanning the image and incrementing the index, which represents an intensity value from 0 to 255, that it correlates with. Afterwards, the cumulative histogram, the running total of all the frequencies in the histogram, is calculated by looping through the histogram and setting each index value to the previous index value plus the current index value. Finally the cumulative histogram values are normalized and are written into the image.



**Figure 3:** Original image and image with Histogram Equalization applied

**3.3 Smoothing**

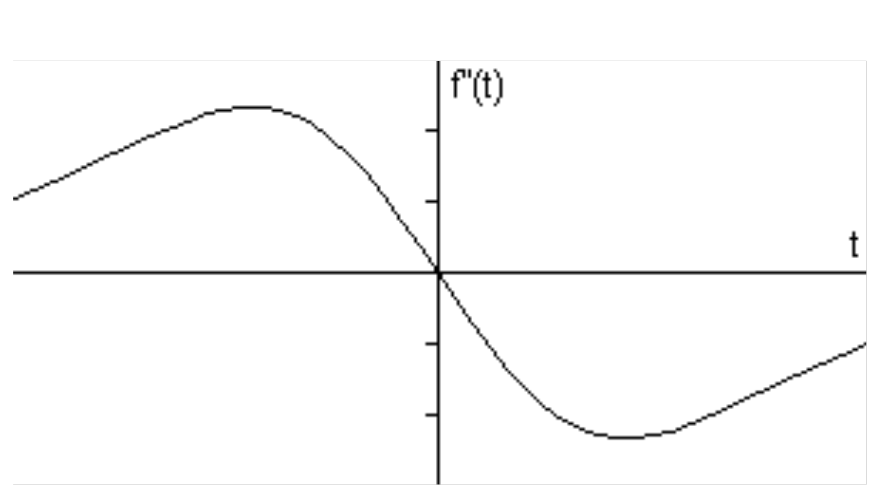
Smoothing an image to homogenize its regions, while reducing noise, is probably the most desirable low-level processing operation to have. In implementation, it is also of utmost importance to preserve edges. Taking this into account, the identification system uses the Median smoothing technique.

In median smoothing, a square “frame” of custom size iterates over the original image. Each pixel brightness value in the frame is collected into an array, and that array is sorted. Then, the median value of that array (the value at the array’s middle index) is output to the corresponding position of an output image (the position is the center of the frame). The frame is then advanced one pixel and the process continues over the entire image. Using a larger sized “frame” results in a more coarse smoothing of the image, because finding the median of a larger area results in a less representative value.

The primary advantage to using median smoothing in particular is that it preserves edges exceptionally well. This intuitively makes sense, because unlike a technique like averaging, median is not swayed by having disproportionate values in the upper half or lower half of the frame values.

**3.4 Edge Detection**

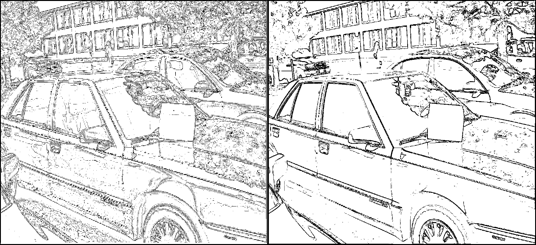
There were two edge detection techniques used within the project. These techniques included using the Laplacian and the Kirsch operators. The Laplacian operator is a differential operator used for identifying edges within an image. The Laplacian is a second order derivative operator that uses the second derivative of images pixel intensities in order to find edges by identifying zero crossings of the Laplacian (Raman Maini, Himanshu Aggarwal).



**Figure 4:** A zero crossing of the Laplacian.

The implementation of the Laplacian edge detection technique goes as follows. A 3x3 kernel was used to identify the zero crossings of the Laplacian by looping over an input image and scanning the kernel throughout the image. Pixel values from the kernel are multiplied with the pixel values from the input image and are then accumulated into a running total. If the sum of these operations is below a certain threshold, the pixel value within the center of the pixels being observed in the input image is set to 255 (white), otherwise it becomes 0 (black). This will result in our final edge image. The Kirsch operator utilizes a far more distinct approach to detecting edges.

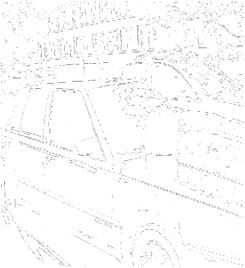
The Kirsch operator is a first derivative operator, unlike the Laplacian operator, rather than looking for a zero crossing, a maximal value is considered to be an indicator that an edge element has been found. The Kirsch operator uses 8 different kernels each of which correspond to a particular direction: N, S, E, W, NW, SW, SE, NE. All 8 kernels are scanned throughout the image producing a value when multiplied with the particular region of an input image. The directional kernel with the maximal value is used to indicate that an edge element has been found.



**Figure 5:** Laplacian (left) and Kirsch (right) edge detection.

**3.5 Thinning**

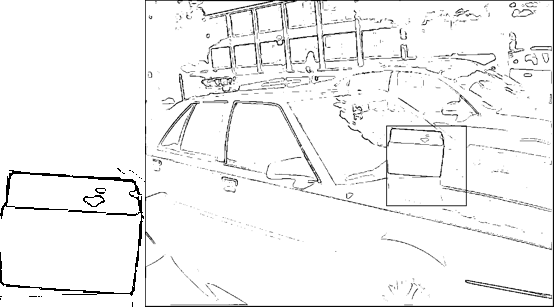
In order to further refine our edge images, a thinning algorithm, invented by Stefanelli and Rosenfeld, was applied (R. Stefanelli, A. Rosenfeld). The main objective of Stefanelli and Rosenfeld’s technique is to retain the medial line/skeleton of an edge image. In order to obtain a thinned image it must be scanned within a loop. At each iteration contour points are collected and removed from the original edge image. Moreover, during each iteration, pixels that are vital to the final result are kept during the process, these are referred to as *final points*. The loop completes once the manipulated edge image and final points are identical.



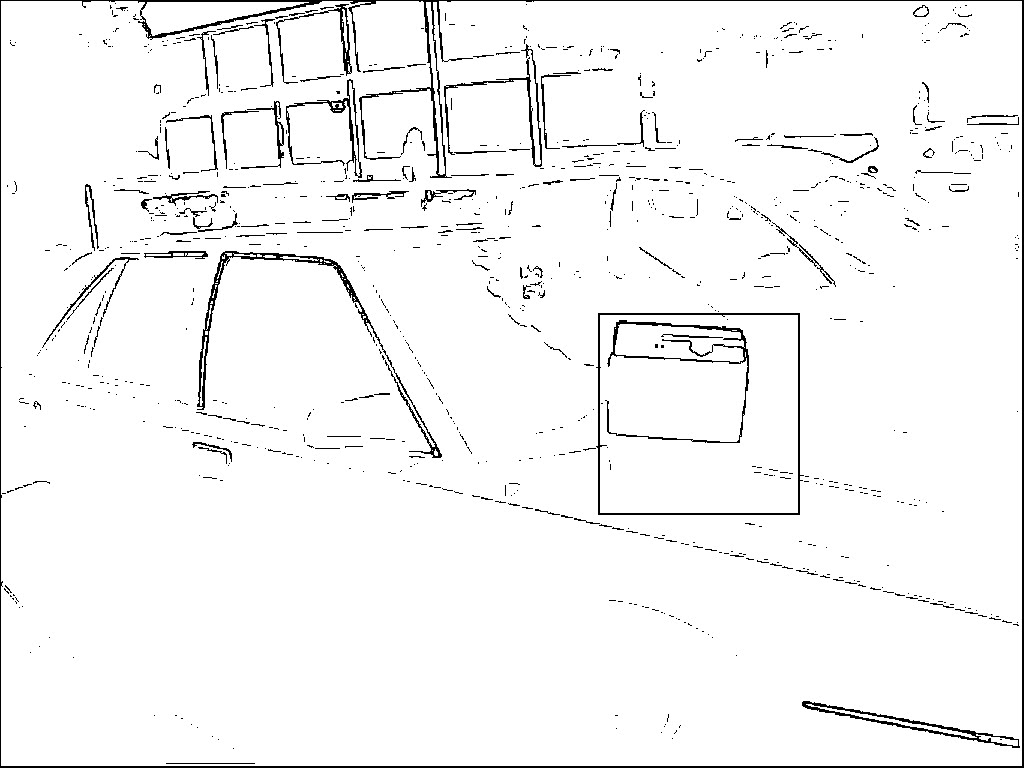
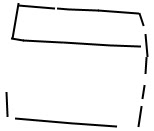
**Figure 6:** Thinning

**3.6 Template Matching**

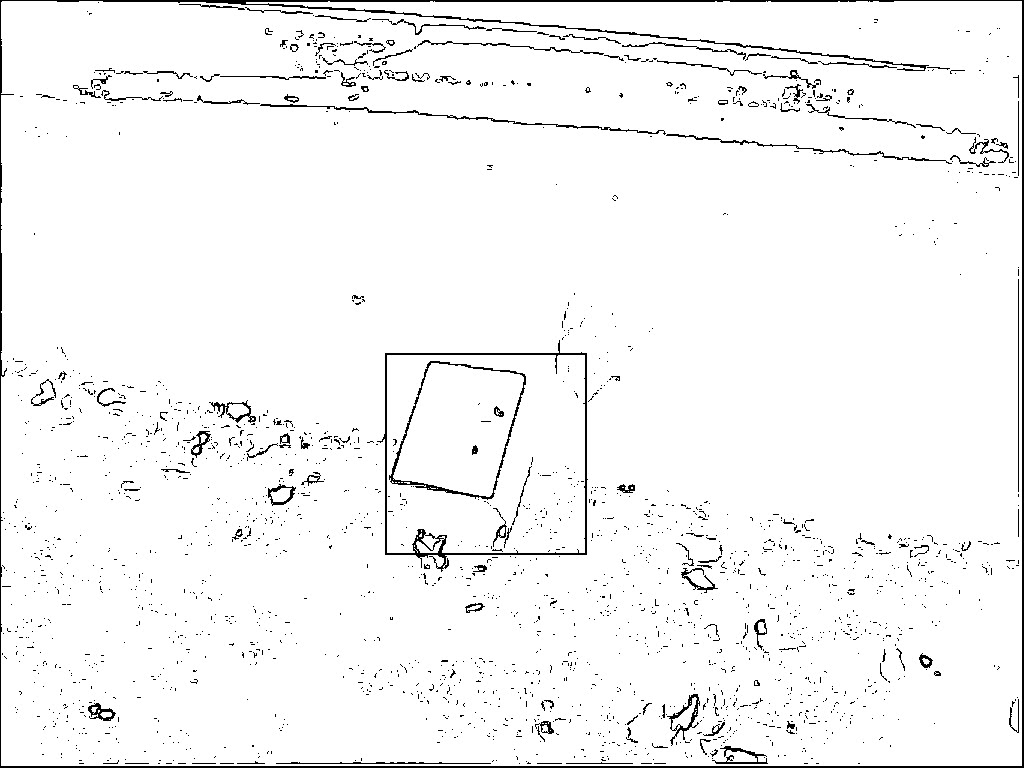
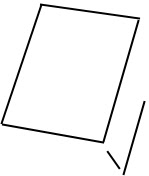
In order to identify a cardboard box within an input image, a template matching algorithm was used. Template matching is the process of looking for specific patterns within an image using a template or specialized kernel. The algorithm for template matching goes as follows (Note that an image containing only edge elements can only be used as input for the algorithm). Firstly, a loop is used to iterate through each pixel of the input image and an inner loop is used to scan a particular template over a particular subset of the image. Secondly, for each iteration we compute the absolute value of the difference between the pixel value from the template and the pixel value of the input image, this is added onto a running total. If the final sum is less than the current smallest sum, the location of the pixel observed in the loop is saved and is considered to be the best possible match to the template.



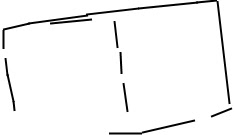
**Figure 6:** The template cropped from the image (left) and the detected box (right).



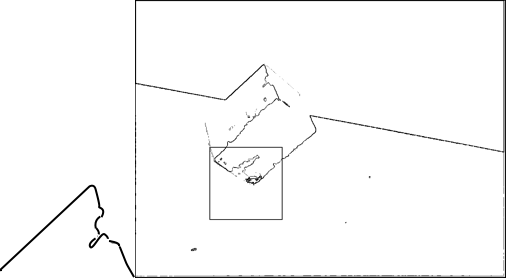
**Figure 7:** A manufactured template (top) and the detected box (bottom), using that template.



**Figure 8:** A manufactured template (top) and the detected box (bottom), using that template.



**Figure 9:** A manufactured template (top) and the detected box (bottom), using that template.



**Figure 10:** A manufactured template (left) and the detected box (right), using that template.

**Template Matching Techniques**

In order to test the effectiveness of the IBVS for cardboard box detection, first, the cardboard box was cropped out of an image. This isolated box was then used as a template to be compared against its own parent image. The desired results can be seen in Figure 6.

After several attempts to make the system more generalized for generic box detection, it was deemed necessary to manufacture custom templates to apply to specific images. Figure 7, 8, and 9 show these manufactured templates. In order to create templates, the processed image was first loaded into the Paint.NET image processing program. Then, a new layer was created on top of the original image. Using the line tool with width of 2, the box in the image was traced, and then the original layer was deleted, leaving only the line drawing with a transparent background. Lines were drawn completely at the discretion of the person creating the template, but of greatest importance was matching the edges and thick lines of the original box image.

**5. Performance**

The performance of an image processing system is paramount, because the amount of data to process in digital images increase greatly with image size. Applying filters requires iterating over the image each time an operation is performed, so improving efficiency in any other possible areas is desirable. Many of the algorithms implemented for the project had a time complexity of N \* M. These algorithms included printing the pixels within an input image, writing data to an image, constructing a 2D array of pixel values, darkening an image, and contrast adjustment. A subset of the algorithms had a higher complexity—- N \* M \* X^2. These algorithms not only looped through each input image (N \* M), but for each iteration performed an operation that had a complexity of X^2, where X^2 represents a separate dimension, repeated in nested loops. These algorithms included Laplacian edge detection, Kirsch edge detection, smoothing, collecting final points for the thinning algorithm, and collecting contour points for the thinning algorithm. The histogram equalization algorithm had a time complexity of N + N^2 and the template matching algorithm has a time complexity of N^2 \* M^2.

**6. Discussion**

As for the overall effectiveness of the system, it is theoretically sound. All algorithms work to their desired purpose, and the template matching function works. However, using accurate and appropriate templates with which to match against the image presented a challenge. Four of seven given objects were able to be detected using manufactured templates, after trial and error.

Performance remains difficult to manage in a system like this, where it is necessary to iterate over each pixel of the image whenever most operations are performed. Despite this, typical execution on an image of size 1024x768 takes a matter of seconds. One possible improvement would be to use images of smaller size, but pre-processing parameters would need to be calibrated again for the different image size.

The potential for future improvement is great, even though the system already works well in its domain. The first order of business is to improve the templates applied to the remaining images. Beyond that, being able to process color images would be a great improvement. Additionally, the effectiveness of pre-processing could be increased. As it stands, it was difficult to isolate just a cardboard box from the given images, as there was lots of undesirable noise left over. Having a way to automatically calibrate pre-processing parameters would be a significant time-saver, as adjusting them by hand consumed a lot of time and effort.

**7. Conclusion**

Designing and implementing a computer vision system to emulate the human visual system is a difficult feat that involves a decent level of understanding of how the brain works in terms of vision. Not only is understanding vision at a biological standpoint necessary to constructing an artificial system, but it also serves as an inspiration in terms of what steps to take in order construct such a system. This inspiration includes the processing of visual information in various stages which forms the basis of our vision and, conversely, the basis of an artificial system. The stages used in an artificial computer visual system includes low-level image processing. The low-level processing used within the project included contrast adjustment, histogram equalization, edge detection techniques, thinning and template matching in order to identify a cardboard box within an image. All of these processes are run at a particular order in order to produce the final result just like how the brain processes signals from the eye to produce what is in our visual field.

**References**

R. Stefanelli and A. Rosenfeld. 1971. Some Parallel Thinning Algorithms for Digital Pictures. *Journal of the ACM* 18, 2 (April 1971), 255–264. DOI:http://dx.doi.org/10.1145/321637.321646

Raman Maini and Himanshu Aggarwal. Study and Comparison of Various Image Edge Detection Techniques. *International Journal of Image Processing (IJIP)* 3, 1.

Richard Szeliski. 2011. *Computer vision: algorithms and applications*, London: Springer.