Scilab case study project on

Analyzing Object Distances in 2-D Images using IPCV toolbox

Fahad Ali

Visvesvaraya National Institute of Technology

July 25, 2024

Abstract

In this research, we present a methodology for computing the distances between the centroids of multiple objects within a 2-D image using the Scilab IPCV toolbox. The approach is divided into two primary phases: segmentation of objects from the background, and calculation of their centroids and inter-object distances. Our program is designed with modular functions to enhance control and readability and supports two operational modes. The auto mode leverages edge detection techniques to automatically identify objects, while the manual mode allows users to iteratively threshold a grayscale image to isolate objects. We address challenges such as effective image segmentation and accurate distance measurement, providing solutions for calculating physical distances by incorporating user-defined scales or reference objects within the image. The efficacy of our approach is validated through comparison with manual measurements, demonstrating its potential for applications in various fields requiring precise object distance computation in images.

1. Problem Statement

The program developed in this case study must:

- Read the input image.
- Segment the image into distinct objects and background.

- Perform various image processing operations to enhance and clean the image.
- Detect the objects and calculate their centroids.
- Compute the distances between the centroids of the identified objects.

Issues Faced

The primary challenge encountered is segmenting the objects from the background. While numerous algorithms exist to address this problem, none offer perfect results. Different algorithms are effective on different types of images, often requiring human intervention for optimal performance. This complexity is further increased by the presence of multiple objects in the image, limiting the selection of suitable algorithms.

To get the distances in mm the user must give the scale of the object in pixels per mm which must be manually computed. This can be done by placing a physical scale in the image and dividing the no of pixels the scale takes up in the image by the unit of measurement. Another way is to place a circular object of a known size in the leftmost part of the image. The program then calculates the scale for you.

2. Basic concepts related to the topic

1. How are images stored

Colored images are created by combining the red, blue, and green colors and varying their intensities to produce any desired color at a specific spot. An image is composed of pixels, which are the smallest units in an image, with each pixel containing RGB colors. By varying the intensities of these colors, different colors are produced. Each color channel is divided based on its intensity level, and the degree of division determines the range of colors. Standard images are 8-bit per channel, providing 256 colors per channel or approximately 1.6 million colors in total. On the lower end, an 8-bit image can represent only 256 colors.

For instance, to represent the color white, the intensity levels of all channels are set to the maximum value (255 for 8-bit images). Conversely, for black, the intensity levels in all channels are at the minimum value (0). These intensity levels of each pixel's color in an image are stored in a matrix form, with rows and columns representing the image's width and height. The value of any element in the matrix indicates the intensity level for that pixel's color channel. Thus, a colored image is stored in three 2-D matrices stacked on top of each other to represent colors. Similarly, black and white images are stored by varying the intensity of the white color.

2. Edge detection

Edge detection is an image processing technique used to identify the boundaries of objects within images by detecting discontinuities in brightness. This technique is essential for image segmentation and data extraction in fields such as image processing, computer vision, and machine vision. Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods. These algorithms help in accurately identifying object edges, which is crucial for various applications and analyses.

3. Canny Edge detection

A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. The DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT from O(N2) N is the data size.

4. Thresholding

An image processing method that creates a bitonal (aka binary) image based on setting a threshold value on the pixel intensity of the original image. The input to a thresholding operation is typically a grayscale or colour image. In the simplest implementation, the output 2 is a binary image representing the segmentation. Black pixels correspond to background and white pixels correspond to foreground (or vice versa). In simple implementations, the segmentation is determined by a single parameter known as the intensity threshold. In a single pass, each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the pixel is set to, say, white in the output. If it is less than the threshold, it is set to black.

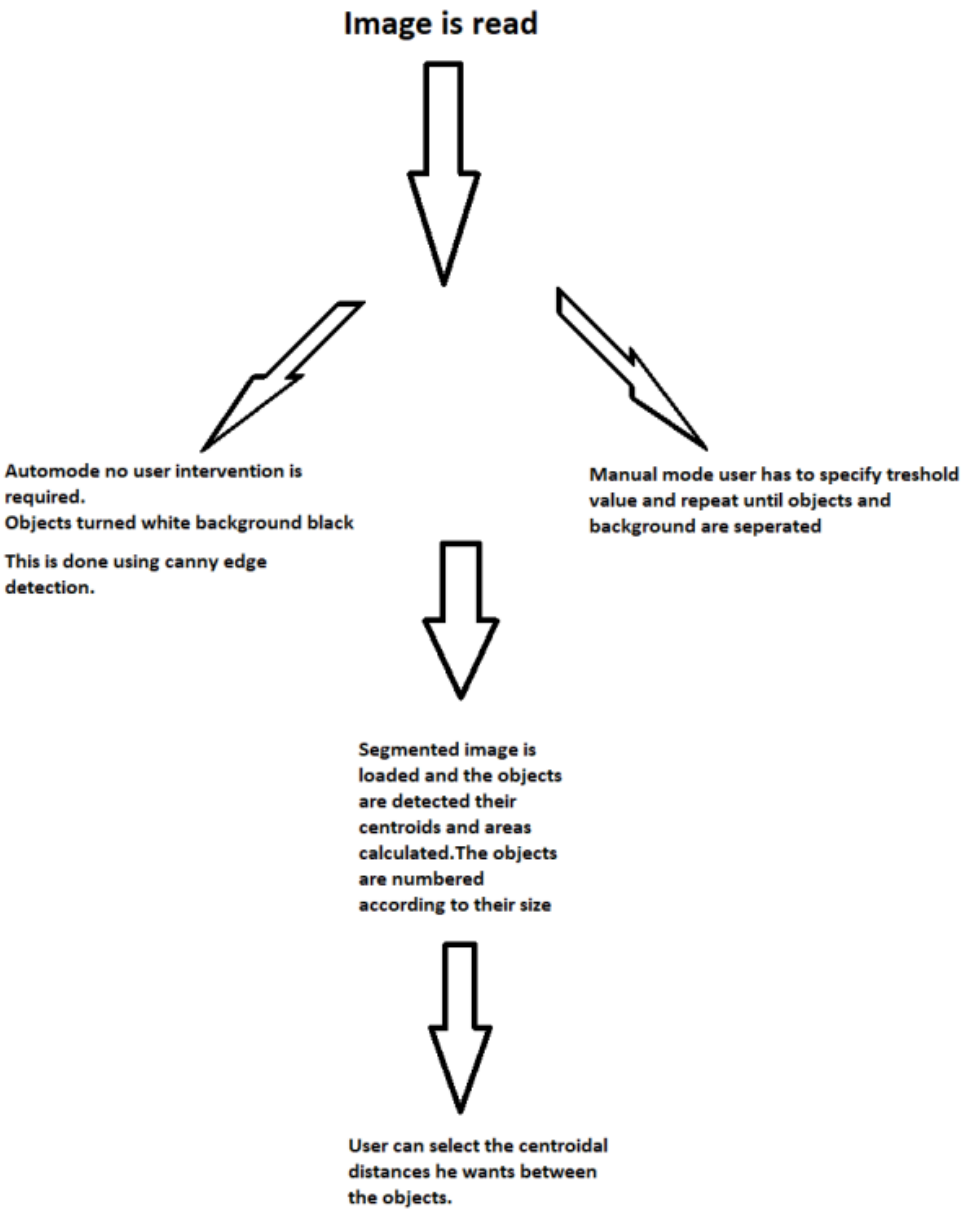
5. Blob Analysis

In computer vision, blob detection methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or colour, compared to surrounding regions. Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered in some sense to be like each other. The most common method for blob detection is convolution.

6. Filtering

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement. Filtering is a neighborhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. Linear filtering is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

3. Flowchart



4. Software/Hardware used

Hardware: Phone camera was used to take sample pictures.

Software: Scilab 2024.1.0 on Windows 10

Toolbox: IPCV – Scilab Image Processing & Computer Vision, version 4.5.0

5. Procedure of execution

(1) First, the main file must be executed ensuring that the Scilab working directory is the same as that of the directory in which the code fi1es are present.

(2) Select the image whose object distances are to be computed from the file explorer.

(3) Select either auto or manual mode some images work with auto mode some with manual.

(4) If auto mode is selected and the segmentation of the image is done properly press enter or else press y to retry.

(5) If manual mode is selected it asks to give a threshold value which gives the best results. It will ask to retry by pressing y or enter to continue.

(6) In manual mode the image takes a lot of time to process as the image is filtered.The time taken also depends on the hardware configuration of the user and can be skipped entirely by pressing the enter key. The results may vary though.

(7) The images are now marked with numbers and are bounded in bounding boxes.

(8) The program asks the user to give the scale of the object if they have placed a circular object on the left it asks for the diameter of the reference object. The user has the option to give the scale which the user can compute if they know the size of the image in pixels and mm. The user has to divide the number of pixels by the size in mm to get scale in pixels/mm. For example if the height of an image in pixels is 400 and height in mm is 100. Then the scale would be S = P ixels ÷ mm That is the scale is S = 400 ÷ 100 = 4

(9) The program asks the user to select the object numbers whose distances are to be computed.

(10) The distance is displayed, which is the distance between the centroids of the object. Further distances can also be displayed as per the user's convenience.

6. Result

1)To validate the results of this case study one can physically measure the distances between the objects and find if the computed distances match with the actual distances. Since physically

measuring the centroidal distances is hard, the horizontal distances were computed. To use the program one can either give the scale or place a circular object of known diameter on the leftmost part of the image this gives a reference object to measure.

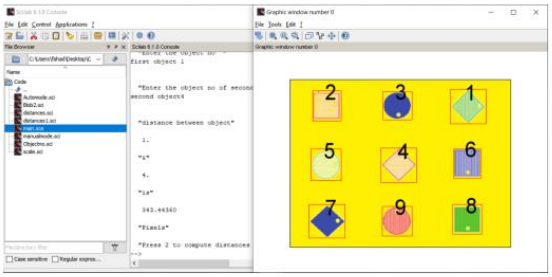
Figure 1: Coins with a horizontal distance of 10 cm

2) In Figure 1 The coins were placed exactly 10 cm apart horizontally. The distance computed by the program was 99.113994 mm which is very close to the actual answer.

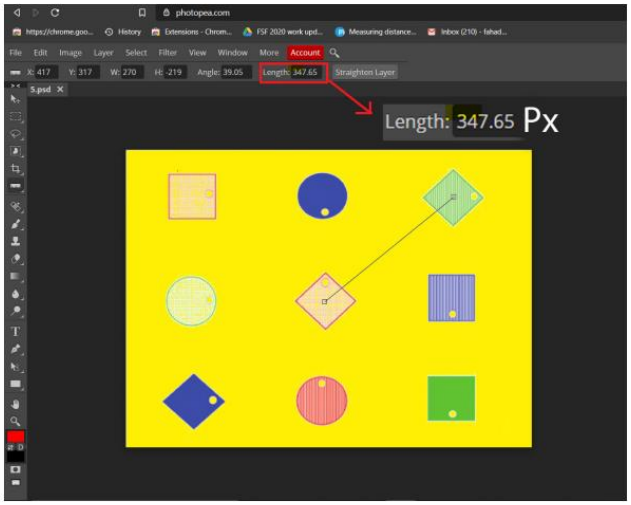
Figure 2: Objects with a horizontal distance of 10 cm

3)In Figure 2, the image on the left is the input image to the program and the image on the right with the scale shows the distance of 10 cm or 100 mm. The distance between the two objects

shown in Scilab is 97.112 mm which is just short of the 100 mm.This error is due to the error in measuring the actual distance which itself can vary by ±5 mm Further the distances computed are the minimum horizontal distances between the objects but the distance measured were from one corner to another which are slightly more than the minimum horizontal distances.



(a) Fig3



(b) Fig4

4)In Figure 3 and 4 the distances were computed in pixels. In Figure 3 the program computes the centroidal distances and displays the output as 343.44360 P ixels. In Figure 4 an external free online program[5](photopea)which computes the distances between two points was used for the validation of the results. The program calculates the distances manually, the user must click on the location of the two points to be measured. The centres of object 4 and 1 were chosen as the two points and the output was 347.65 P ixels which is very close to the value obtained by the program.

7. References

[1] https://scilab.in

[2] https://en.wikipedia.org/wiki/Digital image processing

[3] http://siptoolbox.sourceforge.net

[4] Chapter 4 Object Distance Measurement Using a Single Camera for Robotic Application by Peyman Alizadeh https://zone.biblio.laurentian.ca/bitstream/10219/2458/1/Peyman

[5] https://www.photopea.com