Volume 10 | 2019 Fall 2019

Article 1

2019

Locating Centers of Mass with Image Processing

Amina Gahramanova
University of South Florida

Advisors:
Arcadii Grinshpan, Mathematics and Statistics
Jeffrey Golabek, Mechanical Engineering

Problem Suggested By: Jeffrey Golabek

Recommended Citation
Gahramanova, Amina (2019) "Locating Centers of Mass with Image Processing," *Undergraduate Journal of Mathematical Modeling: One + Two*: Vol. 10: Iss. 1, Article 1.
DOI: https://doi.org/10.5038/2326-3652.10.1.4906
Available at: https://digitalcommons.usf.edu/ujmm/vol10/iss1/1

Locating Centers of Mass with Image Processing

Abstract

The center of mass of a rigid body is a unique point that represents the mean position of all matter that composes it. Allowing representing complex bodies as a single point, this concept is underlying the basis of all essential mechanical calculations, and is therefore a crucial consideration in engineering. The paper shows how to devise a fast and convenient way of locating the centroid of planar objects of different shapes. For several shapes that can be represented as regions bounded by graphs of functions, this point can be found by using integral calculus. However, solving for the center of mass by hand is not only quite tedious, but also inapplicable to objects of arbitrary shape. Thus, it is decided to write a Python-based program that would implement computer vision techniques to scan through images depicting different objects and locate their centroids. For geometric bodies, the results produced by the program are aligned with those predicted by integral calculations. Moreover, the program is also able to successfully mark the centroids of arbitrary shapes, meaning that it is applicable to any possible planar object. The results are then proven experimentally by balancing the analyzed objects about the identified point.

Keywords

center of mass, planar objects of different shapes, Python-based program, computer vision techniques

PROBLEM STATEMENT

The center of mass of a rigid body is a hypothetical point that represents the mean position of all mass distributed within that body. It presents a crucial reference point in mechanical calculations, and consequently, a highly important consideration in different fields of engineering. It is also known as the application point of any uniform force like gravity. The purpose of this paper is to develop a fast and efficient way of locating the centroids of planar objects of different shapes. The first approach is based on manual solving through integration, though it only applies to geometric shapes represented as regions bounded by graphs of functions. The second method, on the other hand, exploits the power of computers, as they can perform similar calculations in a discrete manner. This paper includes a program that locates the centroids of planar objects by implementing image processing and Python programming techniques. The program also has a wider application range, being able to analyze not only geometric bodies, but also objects of any arbitrary shape. The results for both methods are then compared and tested experimentally by balancing 3-D printed objects about the identified points. Arbitrary shapes are also tested experimentally by means of the plumb line test.

Motivation

The centroid of a rigid body is the point of concentration of all its mass, or the weighted average position of all the parts that compose it. This highly useful concept allowing to represent complex bodies as particles plays a large role in simplifying mechanical calculations. For example, Newton's laws of motion, the laws of conservation of mechanical energy, and many other fundamental equations are inapplicable to masses distributed in space. The centroid giving a valid representation of extended objects as point masses allows to analyze them.

The center of mass is also referred to as the center of gravity, since it is the point about which uniform forces produce no torque. Thus, an object pivoted about the centroid is ideally balanced, since gravitational force is concentrated uniquely at that point. It also determines the maximum angle at which an object can be tilted before toppling, since a body is stable only as long as the centroid is directly above the area of support. In short, the use of the center of mass in mechanics and physics is innumerable: it is impossible to calculate linear and angular momenta, define inertial frames, or analyze orbital motion without referencing this point.

Locating the center of mass is particularly important within the field of engineering, which heavily relies on mechanical calculations in design and testing. In civil engineering, for example, the center of mass allows to balance structures with minimum stress, since the load that an object exerts on a structure acts straight below this point. In tilt-slab construction, it is highly important to carefully consider the center of mass when lifting concrete walls to prevent them from cracking (Fig 1).

Figure 1: Tilt-slab construction requires careful consideration of the center of mass (source – intmath.com)

The centroid is especially crucial in aeronautics, since it significantly affects the stability of an aircraft. Operation safety can be ensured only if the center of mass falls within specific limits: ahead of the forward limit makes the vessel less maneuverable, while behind the aft limit reduces stability. Meanwhile for helicopters, the center of mass should always be located directly under the rotor. In biomechanics, the center of mass helps experts in analyzing human locomotion to design better treatments for balance impaired patients. In mechanical engineering, car manufacturers strive to keep the centroid of a vehicle as low as possible to make it more stable in taking turns, which prevents many accidents.

With all these points in mind, one can conclude that the centroid has a wide range of real-life applications and provides useful means of simplifying complex phenomena. Therefore, exploring different ways of locating this point bears significance for many different subfields of engineering

Mathematical Description and Solution Approach

To gain a deeper understanding of what the centroid is, it is useful to first analyze a simple case known as Archimedes Law of the Lever. In this situation, two-point masses m_1 and m_2 are attached to the ends of a thin rod of negligible mass (Fig 2). It can then be experimentally determined that the rod balances about a point for which the following equation holds true:

1. $m_1 d_1 = m_2 d_2$

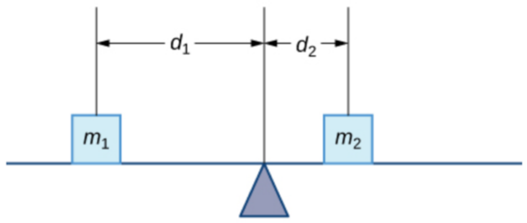


Figure 2: Archimedes Law of the Lever (source – math.libretexts.org)

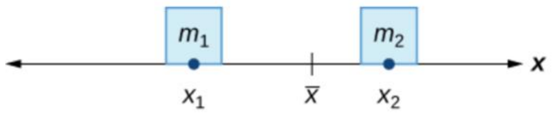


Figure 3: Law of the Lever along a coordinate axis (source – math.libretexts.org)

To translate this finding into a definite coordinate system (Fig 3), one can assume that m_1 lies at the point x_1 along the x axis, m_2 at the point x_2, and the center of mass at the point $\bar{x}$. Relating this to the previous statement, we can see that $d_1 = \bar{x} - x_1$ and $d_2 = x_2 - \bar{x}$. Substituting for d in equation (1) gives:

2. $m_1(\bar{x} - x_1) = m_2(x_2 - \bar{x})$

3. $m_1 \bar{x} + m_2 \bar{x} = m_1 x_1 + m_2 x_2$

4. $\bar{x} = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2}$

To generalize this finding, the center of mass of a system consisting of particles $m_1, m_2, \ldots, m_n$ with coordinates $x_1, x_2, \ldots x_n$ can be represented as equation (5). The quantities $m_1x_1, \ldots, m_nx_n$ are the moments of individual masses with respect to the origin. The system's total moment M_y is equal to the sum of individual moments of each particle. The center of mass itself can be found by dividing the total moment of the system by its total mass (eq. 6). Thus, if all mass were concentrated at the centroid $\bar{x}$, its moment would have been equal to that of the entire system:

$$5. \quad \bar{x} = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i}$$

$$6. \quad \overline{x} = \frac{M_y}{m}$$

For a more general case of a system of particles with two-dimensional coordinates $(x_1, y_1)\ldots(x_n, y_n)$, it is also necessary to account for the moments about the y-axis (Fig 4). In physical terms, the total moments M_x and M_y would measure the tendency of the system to rotate about the y and the x axes respectively. Analogously with the previous case, the moment of the system about the y-axis is equal to the sum of products of individual masses and their distances from the y-axis:

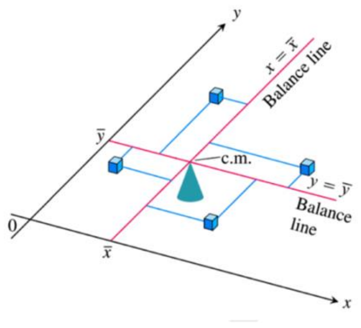
$$7. \quad M_x = \sum_{i=1}^{n} m_i y_i$$

$$8. \quad M_y = \sum_{i=1}^{n} m_i x_i$$

Consequently, the center of mass of a two-dimensional system of particles has to be represented by a pair of coordinates $\overline{x}$ and $\overline{y}$. Just like in the previous case, the product of the centroid coordinates $(\overline{x}, \overline{y})$ and system's total mass gives the total moment about the each of the axes:

$$9. \quad \overline{x} = \frac{M_y}{m}$$

$$10. \quad \overline{y} = \frac{M_x}{m}$$



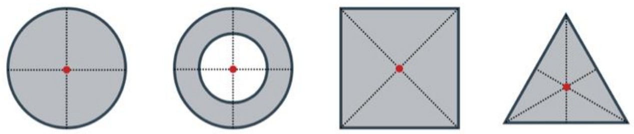


Figure 4: The center of mass of a system of particles in two dimensions (source – math.libretexts.org)

Figure 5: Centroids of common geometric shapes that lie on the symmetry axes (source – khanacademy.org)

When it comes to continuous rigid bodies, however, the approach should differ, since the more complex the object is, the harder to calculate its centroid. While the centers of mass of a few elementary shapes simply correspond to the point of intersection of their symmetry axes (Fig 5), most objects are impossible to analyze with simple algebra. Nevertheless, the same general idea persists: it is still necessary to compute the objects' moments about different axes and their total masses. This can be accomplished by subdividing the object into an "infinite" number of smaller particles and summing them by the means of integral calculus.

The following case considers a lamina, which is an arbitrarily-shaped effectively two-dimensional object with uniform mass distribution. Laminae can be described by the area of the region ***R*** (Fig 6) that they occupy in the x-y coordinate system, and their areal density (mass per unit area) ρ. Considering that the density of the object is constant, the location of the centroid solely depends on the shape of planar region ***R***.

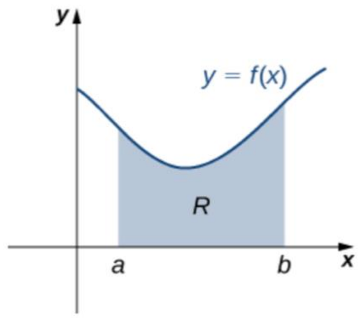


Figure 6: Region R under the graph of function y=f(x) on the interval [a.b] (source – math.libretexts.org)

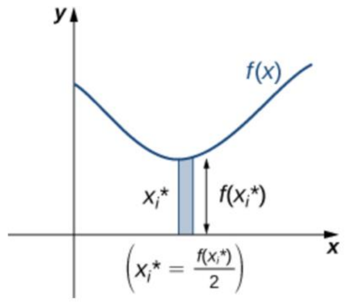


Figure 7: Approximating rectangles with centroids at $\left(x_i, \frac{1}{2}f(x_i)\right)$ *(source – math.libretexts.org)*

First, let us consider the region under a continuous function f(x) bounded by vertical lines x = a and x = b. This continuous region can be subdivided into a number of approximating rectangles of equal width Δx with endpoints $x_0, x_1, \ldots. x_n$. The point x_i^* is the midpoint of each subinterval, defined as $x_i^* = \frac{(x_{i-1}+x_i)}{2}$. Thus, the center of mass of each rectangle lies at its center of symmetry at the point $\left(x_i, \frac{1}{2}f(x_i)\right)$. The mass of each piece is equal to its area times the object's density, that is:

$$11.\ \mathrm{m} = \rho\, f(\overline{x_i})\Delta x$$

The moment of each rectangle about the y axis is then equal to its mass times the distance from its centroid to the axis ($\overline{x_i}$):

$$12.\ M_y = [\rho\, f(\overline{x_i})\Delta x]\ \overline{x_i} = \rho\overline{x_i}\ f(\overline{x_i})\Delta x$$

Taking the limit as n goes to infinity of the Riemann sum of individual moments, we obtain the moment of the entire region about the y axis:

$$13.\ M_y = \lim_{n\to\infty} \sum_{i=1}^{n} \rho \overline{x_i}\, f(\overline{x_i})\Delta x = \rho \int_a^b x f(x) dx$$

Similarly, for finding the moment of the region about the x-axis, we substitute $\overline{x_i}$ with $\frac{1}{2} f(\overline{x_i})$, which corresponds to the distance from the rectangles' centroids to the axis:

$$14.\ M_x = [\rho\, f(\overline{x_i})\Delta x]\, \frac{1}{2} f(\overline{x_i}) = \rho \frac{1}{2} [f(\overline{x_i})]^2 \Delta x$$

Once again, taking the limit of the Riemann sum gives the following expression:

$$15.\ M_x = \lim_{n\to\infty} \sum_{i=1}^{n} \rho \frac{1}{2} [f(\overline{x_i})]^2 \Delta x = \rho \int_a^b \frac{1}{2} [f(\overline{x_i})]^2 dx$$

Just like in the case for a system of particles, the coordinates of the object's centroid are equal to respective moments divided by the total mass. The mass of an effectively two-dimensional object is equal to the product of its density and planar area. Recalling that the area of a region bounded by a graph of function on a closed interval is equal is to the function's definite integral over that interval, the mass of a lamina can be expressed as the following:

$$16.\ m = \rho A = \rho \int_a^b f(x) dx$$

Thus, expanding equations (9) and (10) results in the following expressions for coordinates $(\overline{x}, \overline{y})$ of the center of mass of a planar object. It's important to note that density is canceled in the equation, confirming that the centroid of a lamina depends on its shape only:

$$17.\ \overline{x} = \frac{M_y}{m} = \frac{\rho \int_a^b x f(x) dx}{\rho \int_a^b f(x) dx} = \frac{\int_a^b x f(x) dx}{\int_a^b f(x) dx}$$

$$18.\ \overline{y} = \frac{M_x}{m} = \frac{\rho \int_a^b \frac{1}{2} [f(x)]^2 dx}{\rho \int_a^b f(x) dx} = \frac{\int_a^b \frac{1}{2} [f(x)]^2 dx}{\int_a^b f(x) dx}$$

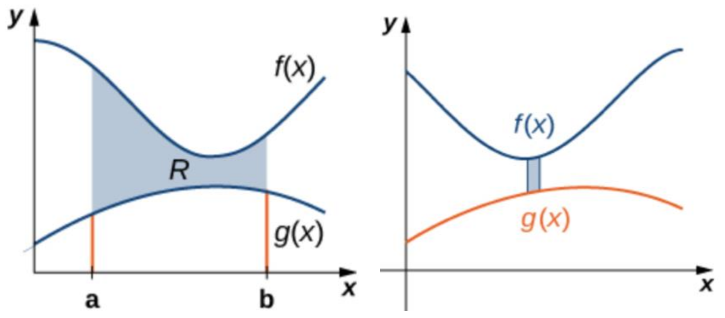


Figure 8: A lamina represented by a region bounded between two curves f(x) and g(x) (source – math.libretexts.org)

Now, let us consider a region bounded by two intersecting curves f(x) and g(x), where f(x)>g(x) on the given interval [a,b] (Fig 8). Using the same reasoning as in the previous case, we can transform equations (17) and (18) as follows:

$$19.\ \overline{y} = \frac{1}{A}\int_a^b \frac{1}{2}\{[f(\mathrm{x})]^2 - [g(\mathrm{x})]^2\}dx$$

$$20.\ \overline{x} = \frac{1}{A}\int_a^b x[f(x) - g(x)]dx$$

Equations (17-20) present an accurate way of finding the exact location of the center of mass of any planar object of uniform density that can be represented as a region bounded by graphs of integrable functions. However, this method is powerless when it comes to planar objects of arbitrary shape that cannot be approximated by functions. Another issue with this approach is that calculating centroids by hand might be quite tedious and error-prone. The objective of this paper is to account for objects of arbitrary shape and explore more efficient alternatives to performing such calculations. This would require relying on the brute force of computers, since they are capable of iterating over thousands of instructions in a matter of mere seconds.

Our approach is to write a code that would analyze an image of a planar object pixel by pixel using for-loop iterations to find the weighted average of the positions of pixels of the same color. In theory, these coordinates should correspond to those of the object's center of mass. The script is written in Python, a high-level programming language that is frequently used to facilitate data analysis in scientific applications. Image processing is performed using Simple CV, an open source framework for building computer vision applications. Computer vision is a programming field that focuses on training computers to analyze and interpret different visuals like images and videos. In this case, the objective is to program a computer in a way that makes it locate the centroid of an arbitrary object captured on a photo and mark it in red.

It would be useful to firstly provide some background on image processing. Digital images can be thought of as giant grids made of numerous small blocks called pixels. The resolution of an image indicates the number of rows and columns in the pixel grid. A resolution of 800×650, for example, indicates that there are 800 columns, 650 rows, and $800 \times 650 = 520{,}000$ pixels in total. Each pixel is defined by a set of three numbers – its RGB values, or the color depth. These numbers usually range from 0 to 255 and indicate the maximum amount of a specific color (red, blue, or green) that would be used within a pixel. For example, a pixel with values (100, 100, 0) would have some red, some blue, and no green. A perfectly white pixel would be defined as (255,255,255), and pitch black – as (0,0,0).

When a picture contains a region of pixels that share a common RGB value (in other words, roughly the same color), it is referred to as a blob (binary large object). It is relatively easy for computers to detect such areas and analyze their dimensions, such as length and width. Thus, if a picture contains an arbitrarily shaped object of uniform color, the computer would be able to treat it as a number of small pieces (pixels) and perform different measurements that would have otherwise been hard to accomplish. This process is similar to integration in the sense that it analyzes larger entities by splitting them into small, simple components.

With that in mind, it is reasonable to assume that computer vision would be able to locate the centroids of objects captured on digital images, given that the color of the object is different from that of the surroundings (so that the computer can distinguish it as a blob). By using a for loop that iterates through each row and column of the image, it would be possible to calculate the moments of each pixel about the x and y axes. Dividing the moments by the total number of pixels in the blob would give the coordinates of the center of mass, which is then marked by a red dot.

DISCUSSION

The first method of locating the centroid explored in this paper uses conventional integration formulae described earlier. Let us examine the region bounded by the graph of function y=2sin (2x) and the x-axis on the interval $[0, \frac{\pi}{2}]$ (Fig 9). Plugging the necessary values into equation (17) results in the point $(\frac{\pi}{4}, \frac{\pi}{4})$, which is approximately equal to (0.785, 0.785).

$$21.\ \overline{x} = \frac{\int_0^{\frac{\pi}{2}} 2x\sin(2x)dx}{\int_0^{\frac{\pi}{2}} 2\sin(2x)dx} = \frac{-x\cos(2x)\Big|_0^{\frac{\pi}{2}} - \int_0^{\frac{\pi}{2}} -\cos(2x)dx}{-\cos(2x)\Big|_0^{\frac{\pi}{2}}} = \frac{\frac{1}{2}\sin(2x) - x\cos(2x)}{-\cos(2x)}\Bigg|_0^{\frac{\pi}{2}}$$

$$= \frac{\frac{1}{2}\sin(\pi) - \frac{\pi}{2}\cos(\pi) - \frac{1}{2}\sin(0) + 0 \times \cos(0)}{-\cos(\pi) + \cos(0)} = \frac{\pi}{4} \approx 0.785$$

$$22.\ \overline{y} = \frac{\int_0^{\frac{\pi}{2}} 2\sin^2(2x)dx}{\int_0^{\frac{\pi}{2}} 2\sin(2x)dx} = \frac{\int_0^{\frac{\pi}{2}} (1-\cos(4x))dx}{-\cos(2x)\Big|_0^{\frac{\pi}{2}}} = \frac{x - \frac{1}{4}\sin(4x)}{-\cos(2x)}\Bigg|_0^{\frac{\pi}{2}}$$

$$= \frac{\frac{\pi}{2} - \frac{1}{4}\sin(2\pi) - 0 - \frac{1}{4}\sin(0)}{-\cos(\pi) + \cos(0)} = \frac{\pi}{4} \approx 0.785$$

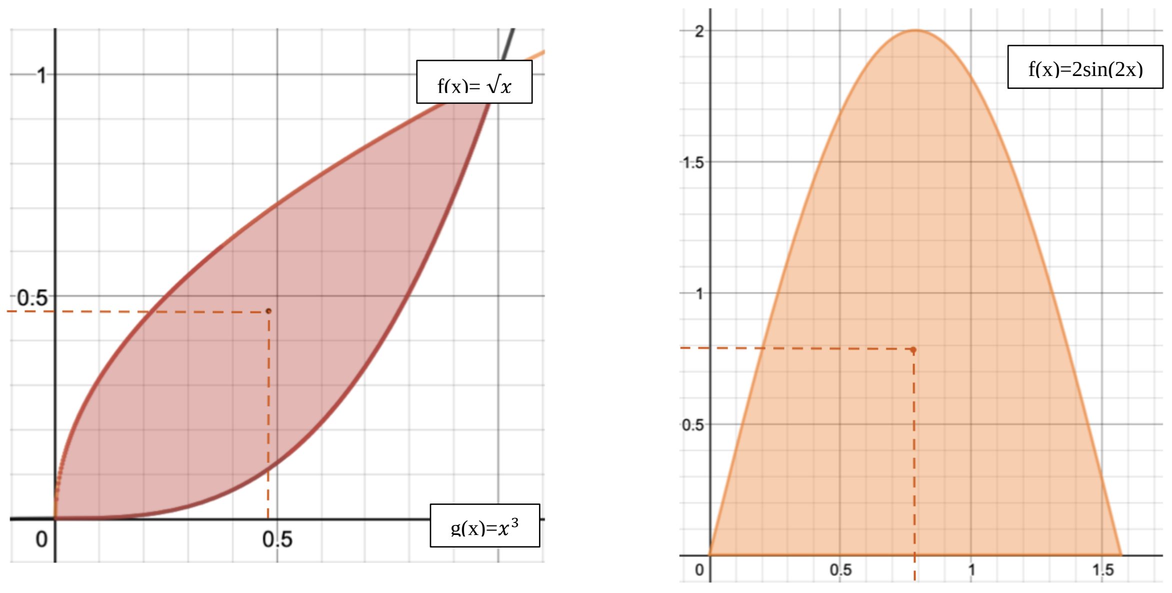


Figure 9: The region bounded by the graph of function f(x)=2sin2x and the x-axis (right) and the region bounded between the graphs of functions $f(x) = \sqrt{x}$ and $g(x) = x^3$(left).

The second investigated lamina (Fig 9) can be represented by the region bounded by the curves $f(x) = \sqrt{x}$ and $g(x) = x^3$on the interval [0, 1]. Substituting for these values in equation (18) shows that the center of mass of this object should be located around the point (0,480, 0.476).

$$23.\ \bar{x} = \frac{\int_0^1 x(\sqrt{x}-x^3)dx}{\int_0^1 (\sqrt{x}-x^3)dx} = \frac{\int_0^1 (x^{\frac{3}{2}}-x^4)dx}{\int_0^1 (x^{\frac{1}{2}}-x^3)dx} = \frac{\frac{2}{5}x^{\frac{5}{2}}-\frac{1}{5}x^5}{\frac{2}{3}x^{\frac{3}{2}}-\frac{1}{4}x^4}\Bigg|_0^1 = \frac{\frac{1}{5}}{\frac{5}{12}} = \frac{12}{25} = 0.480$$

$$24.\ \bar{y} = \frac{\int_0^1 (\sqrt{x}-x^3)^2 dx}{\int_0^1 (\sqrt{x}-x^3)dx} = \frac{\int_0^1 (x-2x^{\frac{7}{2}}+x^6)dx}{\int_0^1 (x^{\frac{1}{2}}-x^3)dx} = \frac{\frac{x^2}{2}-\frac{4}{9}x^{\frac{9}{2}}+\frac{1}{7}x^7}{\frac{2}{3}x^{\frac{3}{2}}-\frac{1}{4}x^4}\Bigg|_0^1 = \frac{\frac{25}{126}}{\frac{5}{12}} = \frac{10}{21} \approx 0.476$$

After obtaining the mathematical values, it is possible to move on to creating a program that would yield identical results momentarily just by analyzing a picture capturing the objects. For this purpose, it is decided to 3-D print physical representations of the shapes of interest, capture them against a dark background, and then run the image processing code upon those pictures (Fig 10). The object on the right is a physical model of the region bounded by the x-axis and the graph of function f(x) =2sin2x, while the other one represents the region between the curves $f(x) = \sqrt{x}$ and $g(x) = x^3$:

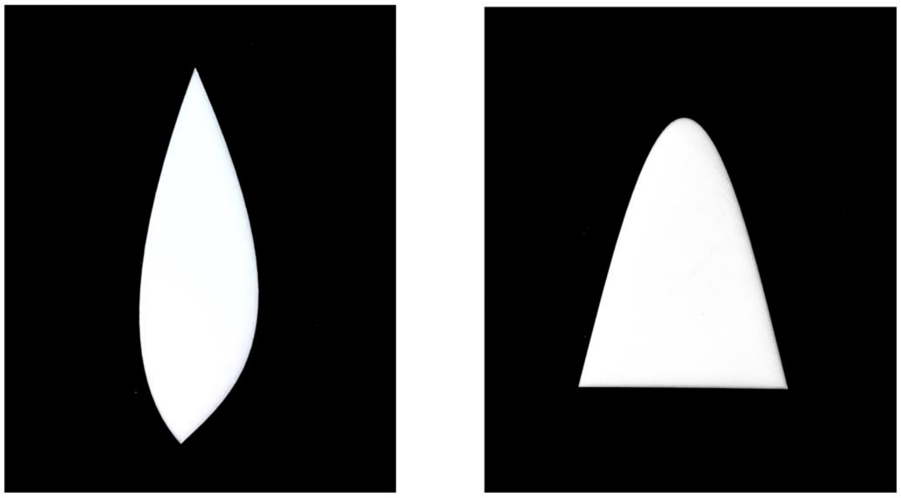


Figure 10: The 3-D printed models of geometric regions bounded by $f(x) = \sqrt{x}$ with $g(x) = x^3$(1) and f(x) =2sin2x (2)

The program itself is written within a Python IDES (integrated development environment software) and makes use of a computer vision library named Simple CV. The full source code for this program can be found in the Appendix. The first lines of the script are meant to import the Simple CV library, which allows for the access and modification of image files. It is necessary to use the file's exact name when assigning an image to the variable in line 3. The fourth line sets the resolution of the imported image to 250×250, although any other value could be used if it is consistent with the boundaries of the for-loops. In this particular example, the resolution is set to 250×250, meaning that there are 250 rows, 250 columns, and 62,500 pixels in total.

```
1) from SimpleCV import Image
3) img = Image("object1.jpg")
4) img = img.scale(250,250)
```

Line 9 indicates the beginning of a nested for loop, which iterates over the columns from 0 to 250 (since this is the number defined by the resolution). The inner loop starts on line 9 b and iterates over all the 250 rows presented. The next lines of the loop's code analyze each pixel for its RGB values. If a pixel's RGB values lie within the white range (i.e. R>225, G>225, B>225), the program recognizes it as a part of the object, and executes the next four lines of code.

```
9) for x in range(0,250):
     a. (R,G,B) = img[x,y]
     b. if (R >= 225 and G >= 225 and B >= 225):
```

Lines i and ii are effectively equivalent to the conventional calculus formulas of finding moments about the x and y axes. Since every image is composed of a set of horizontal units and a set of vertical units, the rows and columns are indexed as a Cartesian coordinate system. Thus, each pixel has a specific pair of coordinates assigned to it, with the x value representing the row, and y – the column. We can think of each pixel as of a point mass; therefore, summing the x and y (row and column) coordinates of each white pixel is identical to adding the moments of each "particle" (white pixel).

```
          i. momentx = momentx + x
         ii. momenty = momenty + y
```

Line iii is the equivalent of the formula for the total mass of the system; each time the program detects a white pixel, it increments the value of the counter, meaning that in the end, it would yield the total number of particles in the object. Line iv is responsible for changing the color of each detected white pixel to blue for making the object more protruding on the resulting image.

```
iii. count = count + 1
 iv. img[x,y] = (0,0,255)
```

Lines 12 and 13 calculate the coordinates of the center of mass, dividing the sum of individual moments of each point mass within the object by the total number of point masses. In basic terms, this formula yields the weighted average of all white pixels in the picture. Finally, line 14 sets the color of the pixel located at the center of mass to red, so that it can be visually detected on the image.

```
12) centx = centx / count
13) centy = centy / count
14) img[centx,centy] = (255,0,0)
```

The program successfully processed the images, marking the location of the center of mass for each object (Fig 11). Coordinate axes are then put over the images to compare the results to those yielded from mathematical calculations.

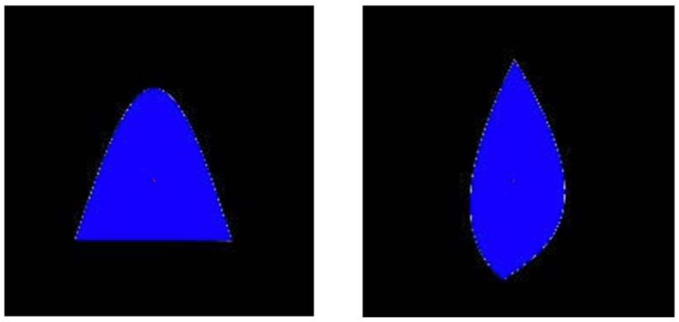


Figure 11: Images after being processed by the program (centroid marked in red).

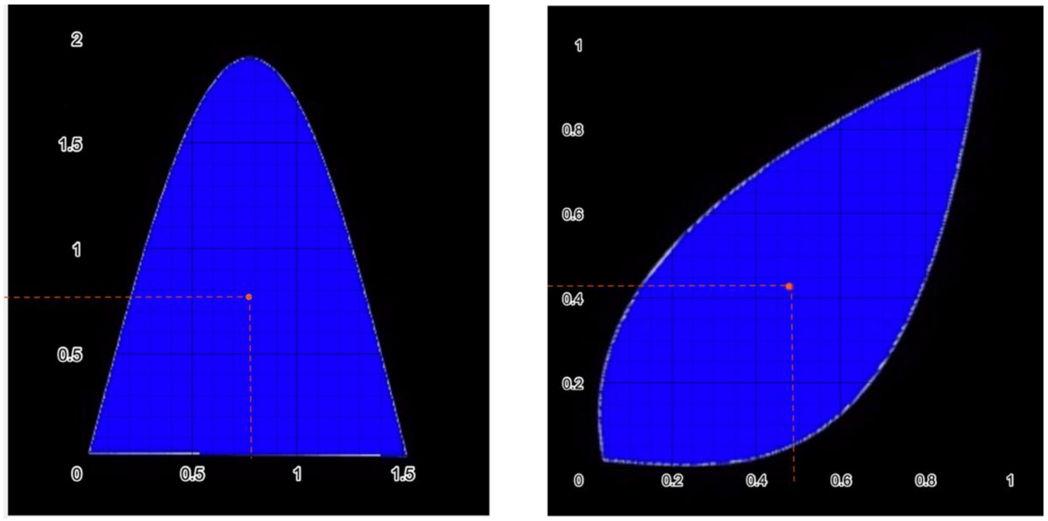


Figure 12: Same images in a coordinate system.

The program locates the centroid of the first object at approximately (0.78, 0.78), which is essentially the same as what is predicted in the earlier mathematical analysis. For the second object, the centroid is laid around the point with coordinates (0.48, 0.48), again consistently with previous calculations (Fig 12).

To test the applicability of these results in real life, it is decided to balance 3-D printed models of these shapes about the points identified above. Since the dimensions of physical objects are not always equivalent to those of graphs, centroid positions are represented as fractions of the objects' transverse and longitudinal dimensions. The centroid of the first shape (2sin2x) is laid at (0.500w, 0.393h), where w and h stand for width and height respectively. For the second region ($f(x) = \sqrt{x}$ and $g(x) = x^3$), this point is located at (0.480w, 0.476h), where h and w correspond to the transverse and longitudinal dimensions of the diagonally placed object. Measuring the first object shows that it is 180mm high and 142 mm wide, meaning that its centroid is laid at (71.0mm, 70.7mm). The w and h of the second one are equal to159mm and

147mm respectively, meaning that the centroid is located at approximately (76.3mm, 70.0mm) as measured from the object's lower left corner (Fig 13).

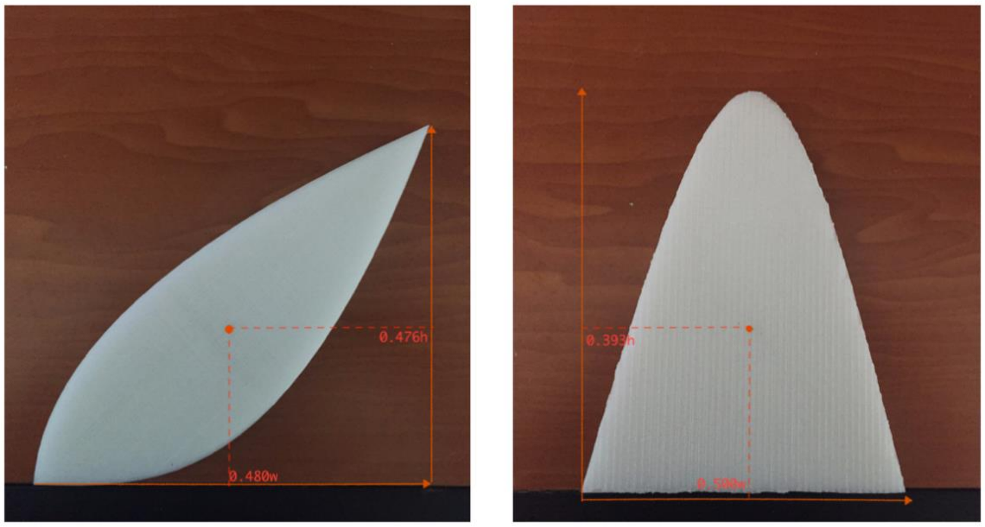


Figure 13: Transferring the obtained centroid coordinated onto physical objects. It can be assumed that the lower left corner represents the origin.

To confirm the findings experimentally, the objects are then pivoted about the marked centroid location. Just as expected, they remain perfectly stable, which means that gravity produces no torque and the center of mass is determined correctly (Fig14).

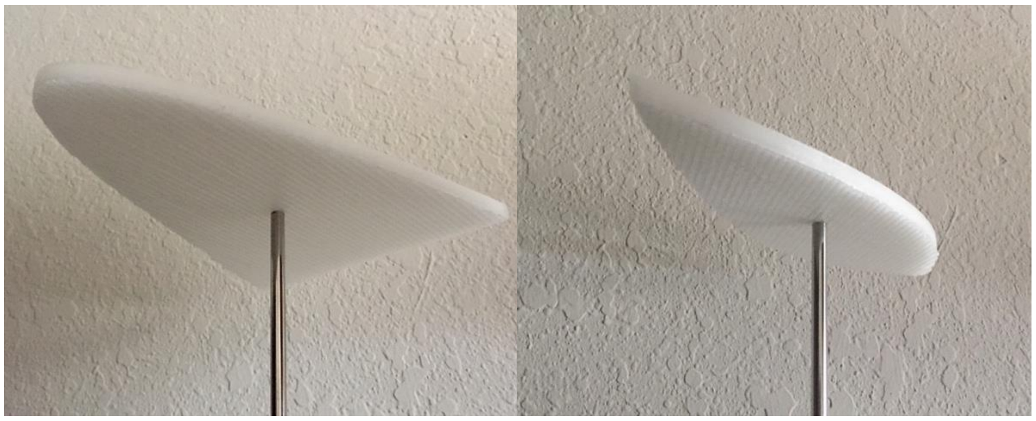


Figure 14: 3-D printed objects balancing about their centroids, further confirming our results.

The program is then run on images of two arbitrarily shaped objects, the centroids of which cannot be calculated with conventional integration formulae. Figure (15) shows the output produced by the program, which once again successfully marks the objects' centroids.

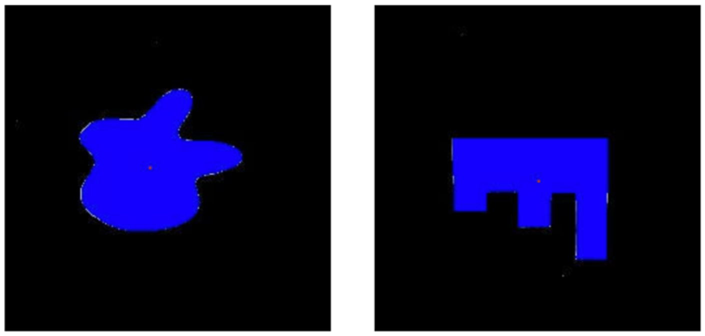


Figure 15: Program's analysis of arbitrary objects, with centroids marked in red.

The validity of these results is then confirmed experimentally by means of the plumb line test. For this purpose, the objects are pinned about a random point and allowed to suspend freely. Due to gravity acting on their centers of mass, the objects rotate until reaching a stable point. There is no torque due to gravity at this equilibrium position, meaning that the centroid must be located right beneath the pin. After a plumb line going down from the pin is drawn, the pin is relocated and the procedure is repeated. The point of intersection of the two lines marks the location of the centroid (Fig 16). The experimental results are consisted with those returned by the program (Fig 15), which further confirms its effectiveness even when it comes to non-geometric shapes.

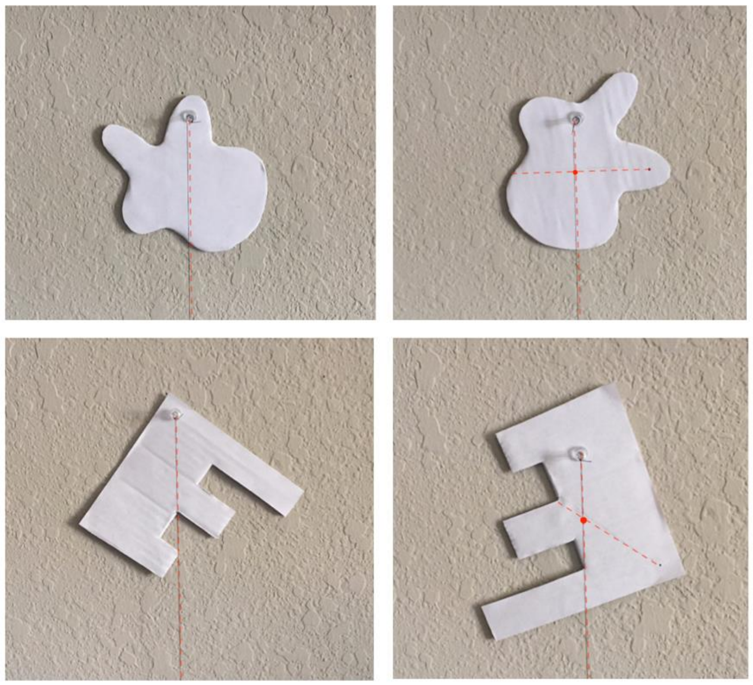


Figure 16: The plumb line test, which allows to experimentally locate the center of mass of arbitrary flat objects.

CONCLUSION AND RECOMMENDATIONS

In summary, we come up with a fast and effective way of calculating the centroids of different planar objects regardless of their shape. This is accomplished by writing a Python-based computer vision program that analyzes digital images depicting the objects of interest. The first step is to calculate the centroids of several geometrically representable shapes using integral calculus. The next step is to physically replicate those shapes, as well as some other arbitrarily shaped objects.

Afterwards we develop a script for a computer program that momentarily analyzes these objects and locates their centroids. The program uses for-loop iterations to scan through the image, detects the pixels comprising the object based on their color, and performs the necessary calculations. For the first two objects, the results yielded by the program are compared to those obtained from mathematical calculations. Just as expected, the points marked by the computer are close to those found by integration (Fig 13). Most importantly, the program also manages to accurately detect the centroids of arbitrary shapes that are too challenging to analyze with regular calculus tools (Fig 15).

Lastly, the validity of results is tested experimentally for both the geometric 3-D-printed models and arbitrary shapes. The geometric objects are pivoted about the centroids obtained from the program and calculations. Just as expected, the object remains perfectly stable, which further reassures us in our measurements (Fig 14). Meanwhile, arbitrary shapes are analyzed using the plumb-line test, in which they are suspended freely about a random point. Comparing experimental results to those returned by the program, it is clear that the points are aligned closely, once again confirming our assumptions (Fig 16).

Thus, it can be concluded that both integration and computer vision are effective in accurately determining the centroids of real-life planar objects. The program has several advantages over the conventional method, being more time-efficient and less error-prone. It also has a much wider range of application, since it is not restricted to geometrically representable shapes only and it can be applied to any planar object of uniform density. In both cases, calculated centroids are applicable to real life objects, allowing to balance them about the identified point.

All slight deviations between mathematically derived centroids and those yielded by the program can be explained by the fact that the 3-D printed objects are not an impeccable representation of geometric curves, partially because of the difference in dimensions. The methodology proposed in this project can be improved by coming up with a more precise way of switching between the pixel, coordinate axes, and physical dimensions, without having to manually put coordinate axes over the images. Another way to improve the program is to expand it to objects with varying density, perhaps by adding more weight to pixels with deeper intensity in the calculations.

NOMENCLATURE

Symbol	Description	Unit
ρ	Areal density	kg/m^2
m	Mass	kg
A	Area	m^2
M_x	Moment about the x axis	$kg \times m$
M_y	Moment about the y axis	$kg \times m$
$\overline{x}$	x-coordinate of the centroid	$-$
$\overline{y}$	y-coordinate of the centroid	$-$

REFERENCES

Books:

Stewart, J. (2012). Essential Calculus: Early Transcendentals. Mason, Ohio: Cengage Learning.

Young, H., Freedman, R (2012). *University Physics with Modern Physics, 14e*. Pearson Learning.

Demaagd K., Oliver A. (2012) *Practical Computer Vision with SimpleCV*. O'Reilly Media.

Web-links:

Mathematics LibreTexts. "Moments and centers of mass" Obtained from: https://math.libretexts.org/Courses/Monroe_Community_College/MTH_211_Calculus_II/Chapter_6%3A_Applications_of_Integration/6.6%3A_Moments_and_Centers_of_Mass

Khan Academy. "What is the center of mass?" Obtained from: https://www.khanacademy.org/science/physics/linear-momentum/center-of-mass/a/what-is-center-of-mass

Biomedical computation review. "Center of Mass Controls Balance" Obtained from: http://biomedicalcomputationreview.org/content/center-mass-controls-balance

APPENDIX

```
from SimpleCV import Image


print "Program Begin"

a = 2*8

img = Image("object1.jpg")

img = img.scale(250,250)

#img.show()

momentx = 0

momenty = 0

count = 0

for y in range(0,250):

   for x in range(0,250):

         (R,G,B) = img[x,y]

         if (R >= 225 and G >= 225 and B >= 225):
                momentx = momentx + x
                momenty = momenty + y
                count = count + 1
                img[x,y] = (0,0,255)

centx = centx / count
centy = centy / count

count = 0

img[centx,centy] = (255,0,0)


print "Program Complete"
```