End of Quarter Report (Summer 2015), Written for Professor Jorge Cortés

Weighted Centroids on Voronoi Cells: Using Finite Elements

Julio Martínez

(Dated: 12 October 2015)

[**Summary**] Current computational algorithms such as Fortune's Algorithm allow for the creation of a Voronoi Diagram (VD) that depends on k number of sites. This report introduces a method to dynamically and iteratively call a VD algorithm and simultaneously compute the weighted centroids of each site's cell within the VD. The dynamic is introduced by proactively moving the sites from their current location to their weighted centroid, upon which their cell boundaries depend. A finite element method with Gaussian Quadrature is introduced to numerically compute a relatively accurate solution. This method allows us to employ nonuniform density functions. This report illustrates the use of a Gaussian distribution.

[Big Picture] This report will walk through the process of developing the Weighted Centroid Voronoi Diagram. A preliminary procedure however is to acquire an algorithm for generating the Voronoi Diagram itself. Here a well known algorithm known as Fortunes Algorithm[1] will be used to gather a dictionary of information for the site indices and their respective cell vertices. All work presented herein was conducted in the MURO labrotaroy at UC San Diego in the department of Mechanical and Aerospace Engineering under the direction of Professor Jorge Cortés, Evan Gravelle, and Aaron Ma.

I. PRELIMINARIES

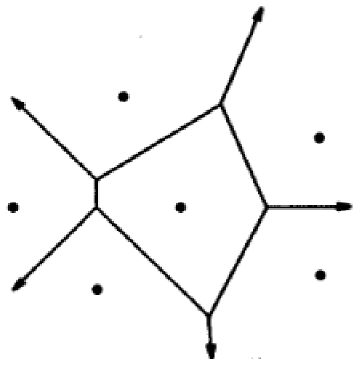
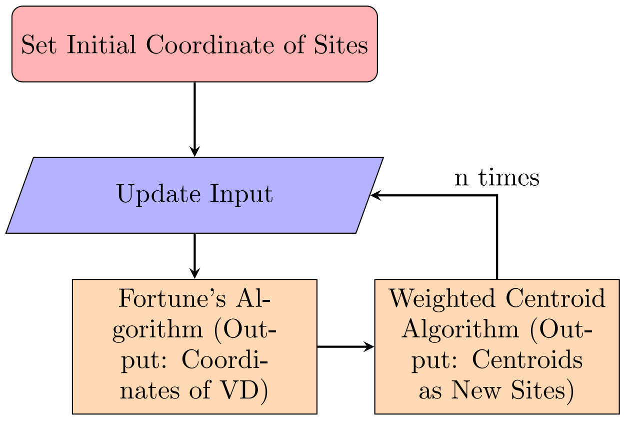


FIG. 1. Voronoi Diagram generated using Fortunes Algorithm on 6 sites[1]

To motivate how the algorithm will be used in conjunction with Fortunes algorithm, the following flowchart illustrates the flow of information. First an initial number of sites are determined and placed as input into Fortunes algorithm. Second, Fortunes algorithm outputs the vertices that correspond to each site's cell boundaries. These vertices along with the the sites used are inputted into the weighted centroid algorithm which outputs the x and y locations of each cell's corresponding center of mass. These centroids are then set as the positions of the new sites. Finally we run Fortunes algorithm one more time to get a VD that depends on the new sites (weighted centroids). This algorithm will eventually converge, however, running n number of times may be more appropriate under time constraints.


Set Initial Coordinate of Sites
Update Input
n times
Fortune's Algorithm (Output: Coordinates of VD)
Weighted Centroid Algorithm (Output: Centroids as New Sites)

A. Finite Element Formulation

The Finite Element Method (FEM) for solving partial differential equations in two dimensions requires evaluating integrals of the form

$$\int\int_e f(x,y)dxdy \tag{1}$$

where e denotes the element. The element is defined by its boundaries or vertices. This report will discuss elements of three nodes (or vertices), i.e triangular elements. Interestingly enough computing the center of mass of any bounded area requires solving the ratio of integrals

$$x_c = \frac{\int\int x\rho(x,y)dxdy}{\int\int \rho(x,y)dxdy} \quad \text{and} \quad y_c = \frac{\int\int y\rho(x,y)dydx}{\int\int \rho(x,y)dydx} \tag{2}$$

Therefore we can utilize the theory of finite elements to numerically compute the integrals found in the numerator and denominator of equation 2. This report will proceed to give an account of the procedure used in formulating a numerical algorithm for computing for the center of mass in a 2-dimensional space.

B. Notation

Consider a 2-dimensional triangular element in with coordinates in (x, y) space. The linear function of the spatial coordinates of any triangle in the parents coordinates x and y is given as:

$$\Theta(x, y) = \alpha_0^e + \alpha_1^e x + \alpha_2^e y \tag{3}$$

Equation 3 can be written in vector notation where $\mathbf{p}(x, y)$ and $\boldsymbol{\alpha}^e$ are defined below.

$$\Theta(x, y) = \alpha_0^e + \alpha_1^e x + \alpha_2^e y = \mathbf{p}(x, y)\boldsymbol{\alpha}^e \tag{4}$$

$$\mathbf{p}(x, y) = \begin{pmatrix} 1 & x & y \end{pmatrix}, \quad \boldsymbol{\alpha}^e = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{5}$$

Now we can express the nodal values $\Theta^e(x_1, y_1) = \Theta_1^e$, $\Theta^e(x_2, y_2) = \Theta_2^e$ *and*, $\Theta^e(x_3, y_3) = \Theta_3^e$ in terms of the parameters α_1^e, α_2^e *and* α_3^e. In matrix notation this is typically written as vector $\mathbf{d}^e$

$$\mathbf{d}^e = \begin{bmatrix} \Theta_1^e \\ \Theta_2^e \\ \Theta_3^e \end{bmatrix} \tag{6}$$

.

Now, we define the matrix $\mathbf{M}^e$ as the matrix whose rows are the corresponding vectors $\mathbf{p}(x_i, y_i)$ for the three given vertices of the triangles (i.e $i = 1, 2, 3$).

$$\mathbf{M}^e = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \tag{7}$$

This allows us to express the system of equations in matrix notation as

$$\mathbf{d}^e = \mathbf{M}^e\boldsymbol{\alpha}^e \iff \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{8}$$

$$\begin{aligned} &\implies \boldsymbol{\alpha}^e = (\mathbf{M}^e)^{-1}\mathbf{d}^e \\ &\implies \Theta^e(x, y) = \mathbf{p}(x, y)(\mathbf{M}^e)^{-1}\mathbf{d}^e \end{aligned} \tag{9}$$

Here we let $\mathbf{N}^e(x, y) = \mathbf{p}(x, y)(\mathbf{M}^e)^{-1}$ denote the element shape function matrix. Note that $\mathbf{N}^e(x, y)$ is a $3x1$ row vector and so we can write $\mathbf{N}^e(x, y) = [N_1^e(x, y) \quad N_2^e(x, y) \quad N_2^e(x, y)]$. Thus there are three shape functions, one corresponding to each node (x_i, y_i) for $i = 1, 2, 3$.

Now we can conveniently express the value at the nodes as

$$\Theta^e(x, y) = \mathbf{N}^e(x, y)\mathbf{d}^e \tag{10}$$

this lead us into the context of isoparametric mapping.

C. Linear Mapping

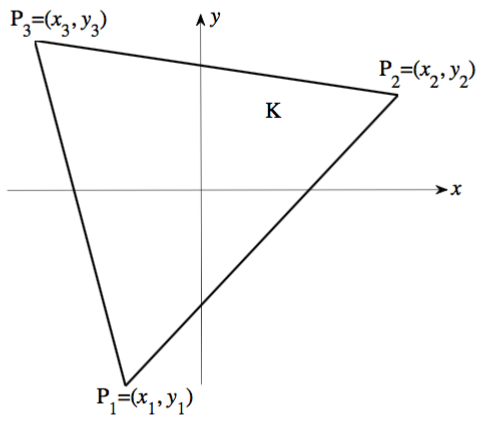


FIG. 2. Triangular element K in (x, y)

Consider the following triangular element K shown in 2. Integrating over this region is easily done analytically. Numerically however it is convenient map these coordinates (x, y) into a more standard region that is well known, in this case a right triangle based at the origin with base and height equal to 1 unit length as show in figure 3. We denote the region of this standard Triangle T_{st} Here we call the (x, y) coordinates the parent coordinates and we introduce the coordinates (ξ, η) to denote local coordinates that we will map to.

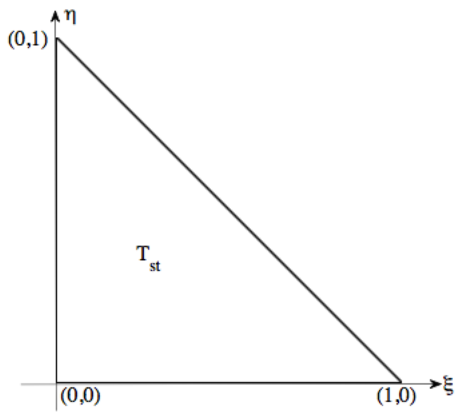


FIG. 3. Triangular element as a right triangle in local coordinates (ξ, η)

We can conveniently create the mapping by using the shape function matrix in the local coordinates (ξ, η) and the nodal coordinates of the parent coordinates (x, y)

$$\begin{aligned} x(\xi,\eta) &= \mathbf{N}^{T_{st}}(\xi,\eta)\mathbf{x} \\ y(\xi,\eta) &= \mathbf{N}^{T_{st}}(\xi,\eta)\mathbf{y} \end{aligned} \tag{11}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad and \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{12}$$

Since in the local coordinates the triangle will always be the same, the shape functions will also always be the same. Solving for these gives:

$$\begin{aligned} N_1^{T_{st}}(\xi,\eta) &= 1-\xi-\eta \\ N_2^{T_{st}}(\xi,\eta) &= \xi \\ N_3^{T_{st}}(\xi,\eta) &= \eta \end{aligned} \tag{13}$$

This leads us to the solution of our integral

$$\int\int_K f(x,y)dxdy = \int\int_{T_{st}} f(x(\xi,\eta),y(\xi,\eta))Jd\xi d\eta \tag{14}$$

where J is the determinant of the Jacobian $\mathbf{J}(\xi,\eta)$.

$$\mathbf{J}(\xi,\eta) = \begin{bmatrix} \frac{dx}{d\xi} & \frac{dy}{d\xi} \\ \frac{dx}{d\eta} & \frac{dy}{d\eta} \end{bmatrix} \tag{15}$$

It can be easily shown that $J = 2A_K$. That is, the determinant of the Jacobian is twice that of the magnitude of the Area of the region in K the triangle (the triangle in the parent coordinates (x, y)).

II. GAUSSIAN QUADRATURE FOR TRIANGULAR ELEMENTS

Finally we are able to reduce our integral into a finite sum using Gaussian Quadrature as follows:

$$\int\int_{T_{st}} f(x(\xi,\eta),y(\xi,\eta))Jd\xi d\eta = \frac{J}{2}\sum_{i=1}^{N_g} w_i f(x(\xi_i,\eta_i),y(\xi_i,\eta_i)) \tag{16}$$

Where N_g is the number of Gauss points needed to approximate N degree polynomials. The gauss points w_i, ξ_i and η_i are conveniently found in textbooks tables. This in total implies that

$$\int\int_K f(x,y)dxdy = \frac{J}{2}\sum_{i=1}^{N_g} w_i f(x(\xi_i,\eta_i),y(\xi_i,\eta_i)). \tag{17}$$

III. METHODOLOGY: FINDING CENTER OF MASS OF EACH TRIANGLE

Now that we have seen that integrating any function $f(x, y)$ as in equation 1 over the domain of a triangular element can be written as a discrete sum, it is easy to see how this can be extended to finding the center of mass of a triangle.

Recall that the center of mass is defined by the ratios in equation 2. Thus the function in the integrand is $f(x, y) = x\rho(x, y)$. Here ρ represents the density distribution over the triangle. Thus the ρ could be a gaussian distribution or exponential etc. Therefore computing the center of mass (x_c, y_c) is is simply an application of the method prescribed above. Therefore we have:

$$\begin{aligned} x_c &= \frac{\int\int_K x\rho(x,y)dxdy}{\int\int \rho(x,y)dxdy} \\ &= \frac{\int\int_{T_{st}} x(\xi,\eta)\rho(x(\xi,\eta),y(\xi,\eta))Jd\xi d\eta}{\int\int_{T_{st}} \rho(x(\xi,\eta),y(\xi,\eta))Jd\xi d\eta} \\ &= \frac{\frac{J}{2}\sum_{i=1}^{N_g} w_i x(\xi,\rho)\rho(x(\xi_i,\eta_i),y(\xi_i,\eta_i))}{\frac{J}{2}\sum_{i=1}^{N_g} w_i \rho(x(\xi_i,\eta_i),y(\xi_i,\eta_i))} \end{aligned} \tag{18}$$

and

$$\begin{aligned} y_c &= \frac{\int\int_K y\rho(x,y)dxdy}{\int\int \rho(x,y)dxdy} \\ &= \frac{\int\int_{T_{st}} y(\xi,\eta)\rho(x(\xi,\eta),y(\xi,\eta))Jd\xi d\eta}{\int\int_{T_{st}} \rho(x(\xi,\eta),y(\xi,\eta))Jd\xi d\eta} \\ &= \frac{\frac{J}{2}\sum_{i=1}^{N_g} w_i y(\xi,\rho)\rho(x(\xi_i,\eta_i),y(\xi_i,\eta_i))}{\frac{J}{2}\sum_{i=1}^{N_g} w_i \rho(x(\xi_i,\eta_i),y(\xi_i,\eta_i))}. \end{aligned} \tag{19}$$

A. Source Code

The source code for computing the center of mass (COM) of any triangle defined by three vertices is listed below. This is written in C++. At first glance Gauss points may not seem enough to approximate a high degree polynomial. However because the ultimate goal of this work is to find the COM of a convex cell created by a Voronoi diagram, then the triangle's domain will only be a small subset of the entire density function. So in essence if we are given a, for example, 20 degree polynomial, we can still get a very good approximation by using a 5th degree polynomial over small domains. Thus the composite approximation will still yield a very small error. The following page contains a source for the the triangle center of mass. Note that the density function may be changed to the density preferred. However for illustration purposes it is set as the gaussian (or normal) distribution with mean u and standard deviation *sigma*.

```
// Gets the COM (x_c,y_c) of triangle
void getTriangleCOM(double &Mass, double &x_c, double &y_c,
                    double X[3], double Y[3], int N_g,double sigma_x,double sigma_y,double u_x,
    double u_y,int rho) {
    //Gauss Info
    double xi[4] = {1.0/3.0,1.0/5.0,1.0/5.0,3.0/5.0};
    double eta[4] = {1.0/3.0,3.0/5.0,1.0/5.0,1.0/5.0};
    double w[4] = {-27.0/48.0,25.0/48.0,25.0/48.0,25.0/48.0};

    //Determinant of J
    double J = ((X[1]-X[0])*(Y[2]-Y[0])-(X[2]-X[0])*(Y[1]-Y[0]));
    std::cout << "J = " << J << std::endl;

    //Declare/Initialize loop variables
    double E_x =0; double E_y = 0; double x,y,DENSITY;
    Mass = 0;

    for (int i = 0; i < N_g; i++){
        //Mapping
        x = ( 1-xi[i]-eta[i] )*X[0] + xi[i]*X[1] + eta[i]*X[2];
        y = ( 1-xi[i]-eta[i] )*Y[0] + xi[i]*Y[1] + eta[i]*Y[2];


        //Density Function!!!!!
        //||-----------------||
        DENSITY = (1/(2*PI*sigma_x*sigma_y*sqrt(1-rho*rho)))
        *exp(-1/(2*(1-rho*rho))*((x-u_x)*(x-u_x)/(sigma_x*sigma_x)
        + (y-u_y)*(y-u_y)/(sigma_y*sigma_y)
        - 2*rho*(x-u_x)*(y-u_x)/(sigma_x*sigma_y)) ;
        //||-----------------||

        //Apriori Expected Values and Mass
        E_x = E_x + w[i]*x*DENSITY;
        E_y = E_y + w[i]*y*DENSITY;
        Mass = Mass + w[i]*DENSITY;
    }

    //Expected Values and Mass With Jacobian
    E_x = (J/2.0)*E_x; E_y = (J/2.0)*E_y;
    Mass = (J/2.0)*Mass;

    //Center of Mass
    x_c = E_x/Mass; y_c = E_y/Mass;
}
```

IV. CENTER OF MASS FOR CELLS

As can be see in the code after the center of mass for each triangle is computed we simply compute the composite center of mass of all triangles within a specified cell in order to determine the weighted centroid of the cell itself. This is done using the following discrete formula:

$$\begin{aligned} Mass_{tot} &= \sum_{i=1}^{n} Mass_i \\ x_{centroid} &= \frac{1}{Mass_{tot}} \sum_{i=1}^{n} x_{c_i} Mass_i \\ y_{centroid} &= \frac{1}{Mass_{tot}} \sum_{i=1}^{n} y_{c_i} Mass_i \end{aligned} \tag{20}$$

.

This turns out to be one iteration of the weighted centroid algorithm. This computation must be iterated for all cells. Thus the number of agents gives an estimate to the computational expense. A report with these details will be given in my End of Quarter Report for Fall of 2015. After all these locations are determined they are collected by placing them into an array. This information has already been reported by Bruno Maciel in his end-of-quarter report. However, I was involved in this part of the project in great detail. Various aspects on the source code are found in Bruno's End of Quarter report for Summer of 2015 and listed below as[3].

In addition much of the finite element theory and notation above was derived using the Thomas J.R. Hughes's textbook on Finite Element Method. This is also cited below as[2].

V. TIPS

It is important to understand that the theory for Gaussian Quadrature was developed exactly for polynomials. In other words, Gaussian quadrature will provide exact results for polynomial integration when implementing the right number of interpolation points or nodes. This however does not hold generally and so will only give approximate values for the Gaussian distribution, which is the distribution of choice for the coverage control algorithm. Because of this it is important to keep in mind that a higher number of nodes will produce better results. These points can either be solved for using basis functions or they can be looked up in tables commonly found in textbooks.

The current implementation does not have a general way of setting the density function and as such is limited to the Gaussian distribution or some other distribution so long as it is hard coded as shown in the source code above. A possible improvement would be to generalize this by inputting interpolation points. Then, by running an interpolation algorithm such as divided differences one may retrieve the polynomial defined by those points. This would allow the user to input any function he or she desires.

VI. PITFALLS

The reason the Finite Element method was chosen to solve the weighted centroid problem was natural. At first, an idea to use polar coordinates was proposed as a viable solution. However, this became problematic. In order to use polar coordinates the cell would still need to be divided intro triangles as done before. At this point the radius of each triangle with respect to the agents location and the boundary of the cell that forms the hypotenuse of the triangle is a function where the radius changes with the angle θ. Keeping r constant would simplify the problem, however, this would grossly approximate the answer in many cases leading to large errors. On the other hand if the function for the radius was kept exact, then the limits of integration over the variable for the radius, r, would be functions of θ leading to unnecessary complexity in the numerical methods employed. This can be imagined by taking any triangle and using polar coordinates over the triangle. This approach was eventually discarded. In coming up with a solution, rectangular coordinates were reckoned a more natural environment for triangles. Thus to keep the problem simple, elegant, and robust a natural use of triangular finite elements was more than appropriate for this problem.

Another problem (this quarter) we have been running into with the computations is a small perturbation or movement in the centroid solutions which exposes itself in the deployment of robots as "jittering." This problem is exaggerated when the standard deviation of the Gaussian distribution is small. Changes were made to the code by adding a constant greater than zero to the density of Gaussian function to avoid situations that divide by zero (in case of round off errors). In addition a "lag" term was added to mitigate a lag in centroid finding. This all encompasses the current work taking place with the centroid algorithm. More details on this work will be provided in the End of Quarter Report for Fall of 2015.

VII. CONCLUSION AND FUTURE WORK

The weighted centroid algorithm takes advantage of the diagram produced by running Fortune's Algorithm. It uses a finite element method with triangular elements to solve the weighted centroid, which is expressed as a ratio of integrals. These centroids are then used to produce a position toward which the agents should move. Once the agents have moved Fortunes Algorithm is ran again using the newly publish positions. Eventually this process converges. In conclusion, the finite element method proved to be a useful tool in the computation of the cells center of mass or as is named in this report "weighted centroids."

Our future plans for this quarter are to eliminate the jittering problem. In addition, we ill introduce multiple Gaussian distributions. Currently a simple addition of a Gaussian distribution may not prove to be useful since all agents may collect towards the nearest Gaussian. However, our vision for this multiple Gaussian problem is to distribute the agents in such a way that all distributions are used effectively. My addition personal plans are in involving myself in the learning of android application to help in the testing and control of the turtleBots and quadrotors. I will also continue to make updates to the website whenever needed.

[1]Fortune, Steve. AT&T Bell Laboratories. *A Sweepline Algorithm for Voronoi Diagrams.* ACM New York, NY, 1984.

[2]J,R. Hughes, Thomas. Dover Publications Inc. *The Finite Element Method: Linear Static Dynamic Finite Element Analysis* Mineola, New York, 1987.

[3]Bruno Maciel. End of Quarter Report *Weighted Centroid Computation and Different Boundaries Shapes* Summer 2015.