The ellipse fitting is one of the most commonly used fittings for pattern recognition and computer vision. However, one of the issues lies in the linear least squares solution, with the conic general equation:

$$F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{1}$$

Where the parameters a,b,c,d,e, and f are linear. The polynomial distance F(x,y) is called the algebraic distance of any point (x,y) from the conic. One of the major problems is that the algorithm will most of the time find a way for all the parameters to be zero. For this to be prevented, the parameters must be constrained. The second problem is that it is not guaranteed that the conic will be able to fit an ellipse. For the conic to represent to represent an ellipse, the parameters must satisfy (2);

$$b^2 - 4ac < 0 \tag{2}$$

In the reviewed paper, "Direct Least Squares Fitting to an Ellipse", it demonstrates taking the parameter of vector a, and scaling it to impose the equality constraint:

$$4ac - b^2 = 1 \tag{3}$$

The quadratic constraint can be expressed in the constraint matrix form C (4) to help solve the solution for the generalized eigenvalue problem,

$$C = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$Sa = \lambda Ca \tag{5}$$

Where S is the scatter matrix DT * D.

The objective of this project is to implement the ellipse fitting onto an FPGA board using Vitis HLS. The purpose will be to compute the ellipse fitting over shock/vibration data generated to detect abnormalities. When first exploring multiple ways to implement the fitting, one of the first methods, the Fitzgibbon method.

This method essentially fits an ellipse to a set of scattered 2D data points(e.g., vibration data). However, this method has proven to be unreliable for hardware use due to how expensive it is to use on hardware. To resolve this, other decomposition methods were considered to replaced the Fitzgibbon method and help solve the general eigenvalue problem, while still being least expensive on the hardware.

The LDLT factorization is a decomposition method that is closely related to the classical Cholesky factorization. Where L is the lower triangular matrix, D is the diagonal matrix, LT is the upper triangular matrix. This can be written as:

$$S = LDL^T \tag{6}$$

Although the first idea would be to consider using the Cholesky factorization instead of the LDLT factorization, this has proven to be ineffective when it comes to being used for the ellipse fitting. The Cholesky factorization requires the given symmetric matrix to be positive definite, meaning that all the eigenvalues must be positive. The LDLT factorization is more flexible with indefinite matrices since it allows the diagonal matrix to have negative or zero entries. In order to use the LDLT factorization to find the solution for the generalized eigenvalue problem (7), the constraint matrix C must be factorized (8):

$$Sa = \lambda Ca \tag{7}$$

$$C = LDL^T \tag{8}$$

This is done to reduce the generalized eigenvalue problem (7) into a standard symmetric eigenvalue problem, where the LDLT factorization is used to set C as (9) using the LAPACK Users Guide Third Edition (Generalized Symmetric Definite Eigenproblems.)

$$Sa = \lambda LDL^T a \tag{9}$$

Furthermore, the right-hand side of equation (7) is then simplified to remove the triangular matrices L and LT. The inverse of L is then multiplied by on both sides of the equation to help simplify the equation into the standard eigenvalue problem

$$L^{-1}Sa = \lambda(L^{-1}LDL^T)a \tag{10}$$

which can be reduced to:

$$L^{-1}Sa = \lambda DL^T a \tag{11}$$

A new variable (12) is then substituted to recover the eigenvectors of variable a of the original problem.

$$z = L^T a => (L^{-T})z = a => a = L^{-T}z \tag{12}$$

Equation (11) is then taken and a to z to become:

$$L^{-1}S(L^{-1}z) = \lambda DL^T(L^{-1}z) \tag{13}$$

Which is reduced to:

$$L^{-1}SL^{-T}z = \lambda Dz \tag{14}$$

(11) is then normalizes, which is done by multiplying both sides of the equation by the inverse of D, so that we get left with equation (15)

$$\lambda z \tag{15}$$

(lambda are the eigenvalues of the standard eigenvalue problem, and z is the eigenvectors, which will correspond to the ellipse solution).

$$(D^{-1}L^{-1})SL^{-T}z = \lambda(D^{-1}D)z \tag{16}$$

which is reduced to:

$$(D^{-1}L^{-1})SL^{-T}z = \lambda z \tag{17}$$

Which can also be written as

$$Az = \lambda z \tag{18}$$

With the new standard symmetric eigenvalue problem, the eigenvalues and the eigenvectors of (18) is solved. The matrix $A = D^{-1}L^{-1}SL^{-T}$ is symmetric, and therefore can be diagonalized using an orthogonal transformation. To compute its eigenvalues and eigenvectors efficiently on hardware, the cyclic Jacobi method is implemented. These transformations preserve the symmetry and the eigenstructure of A at each iteration, ensuring accurate eigenvalues and eigenvectors even under limited numerical precision. Also, since it uses only simple and fixed rotations, it becomes hardware-friendly for the FPGA board.
Initially, the Jacobi method takes the largest off-diagonal of the symmetric matrix Apq. The two indices (p,q) identify the plane of rotation, which means the Jacobi method applies a rotation angle in the (p,q) plane, which makes Apq = 0.

$$(p, q) = \arg\max_{p<q} |A_{pq}| \tag{19}$$

$$\tan(2\theta) = \frac{2A_{pq}}{A_{qq} - A_{pp}} \tag{20}$$

$$\text{t} = \begin{cases} \frac{1}{\tau+\sqrt{1+\tau^2}}, & \tau \geq 0, \\ -\frac{1}{-\tau+\sqrt{1+\tau^2}}, & \tau < 0, \end{cases} \quad \text{where } \tau = \frac{A_{qq}-A_{pp}}{2A_{pq}}.$$

For the rotation in the (p, q) plane, only the p-th and q-th rows and columns of A are modified. The updated diagonal and off-diagonal elements are computed as:

$$A'_{pp} = c^2 A_{pp} - 2scA_{pq} + s^2 A_{qq}, \tag{21}$$

$$A'_{qq} = s^2 A_{pp} + 2scA_{pq} + c^2 A_{qq}, \tag{22}$$

$$A'_{pq} = A'_{qp} = 0, \tag{23}$$

and for all $i \neq p, q$:

$$A'_{ip} = cA_{ip} - sA_{iq}, \tag{24}$$

$$A'_{iq} = sA_{ip} + cA_{iq}. \tag{25}$$

These updates maintain the symmetry of the matrix after each rotation.