* Huffman coding, calculate probabilities of each symbol appearing, then form a tree by combining the symbols that appear least until there is only 2 branches. Then assign codes by adding a 1 to the symbol with lower appearance and 0 to the other
* Avg Huffman code length is P(Ei)\*code length.
* Optimal code length/efficiency is equal to message length times entropy(

|  |  |
| --- | --- |
|  |  |

* Numeric coding, calculate probabilities of each message, set them on a scale, then for each successive part of the message, replicate the scale at the width of the value

Chart, scatter chart

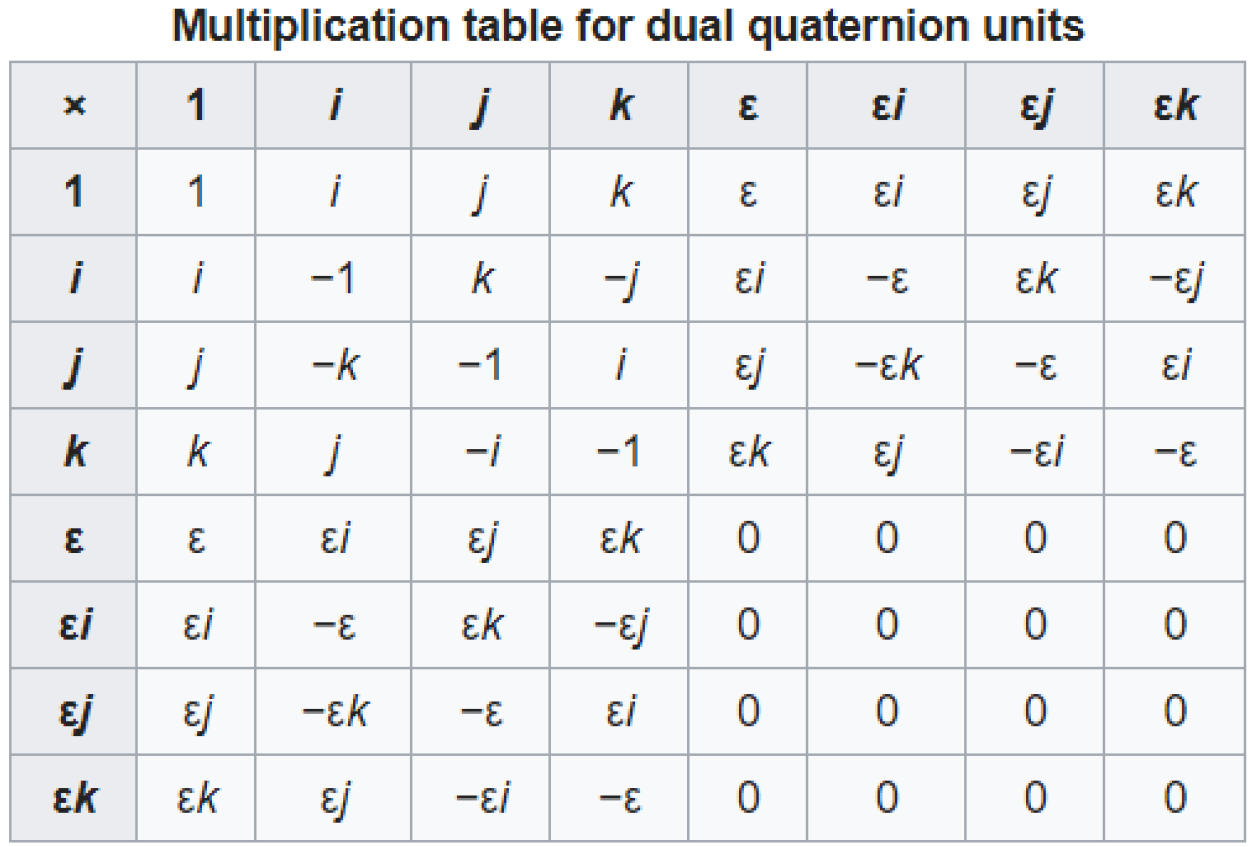
Description automatically generated

* To represent a binary number with an interval for example 0101, write as a decimal 0.0101. Then for a lower bound add recurring 0 after it 0.01010, and for the upper bound, add recurring 1 0.01011. Then add by the powers of 2. For the upper bound, use a value of 1 instead of the last 0 that appears before the recurring 1
* A picture containing text

  Description automatically generated
* To find a good, encoded range, convert upper and lower bounds to binary. Truncate to code length and add or remove 1 until the truncated binary is between upper and lower bounds
* Transformation matrices are ordered right to left, last to first. I.e. if T is applied first then R, then the order is RT
* Angle between two vectors v = (a, b, c) and u = (x, y, z),
* Direction normal to two vectors v = (a, b, c) and u = (x, y, z),
* Homogenous matrix to rotate by an angle around an arbitrary normalized direction (x,y,z)
* Text

  Description automatically generated with medium confidence

|  |  |
| --- | --- |
|  |  |

* PCA analysis, calculate mean of all the points, and subtract from the original points to center the values.
* Calculate the covariance matrix by multiplying the matrix of centered points to it’s transpose
* Subtract 𝜆I from the covariance matrix and calculate the determinant of the result. This will lead to a quadratic equation, so find the two possible values of 𝜆
* Then calculate the eigen vectors using the two values of 𝜆 and the following constraints:
* Distortion(k-means) is the squared sum of all points to their closest final cluster center
* Motion capture data is stored as a skeleton file. The root point has a translation vector, the rest has rotation.
* Dual quaternion replaces a and b in a dual number with quaternion each. One for orientation, one for translation.
*  let d1 = a + be and d2 = c + de, d1.d2 = ac + ade + cbe (use distribution)
* In a dual quaternion, if a = identity, translation only. Conversely if be is multiplied by 0, rotation only
* To apply a quaternion to a vertex, use v`=qvq\*, where q\* is the conjugate of q(change second part to negative)
* Valence driven coding: store number of vertices each vertex is connected to, perform Huffman coding on totals.
* Decimation of △ meshes: if a non-complex vertex is within a certain distance to avg plane of surrounding △s, its deleted. Then when only 3 vertices left, use a split line that reforms triangle based on smallest aspect ratio
* Geometric optimization: take 3 vertices, if v0-2 can connect without intersection or out of bounds, del v1. Else new set of 3 from v1.
* Simplification envelopes: form 2 layers around mesh, reduce points without intersecting either
* Convex hull: border such that any points can be connected without intersecting the hull.
* Voronoi diagram: lines such that they are perpendicular bisectors of lines that connect points
* Progressive mesh: combine/collapse edges to simplify meshes based on ΔE(E(M) = Edist(M) + Espring(M) +Escalar(M) + Edisc(M))

