JPEG Recovery Algorithm

Author: CD Jones

Revision: 1.0

**Algorithm Phases**

**Introduction**

Since the data in the file matrix in question can be corrupt in multiple places, it may not be loaded hierarchically to start. Initially, it must be treated sequentially, run through various **validation** passes, and then the data must be transferred into a temporary instance of a JPEG File Tree Structure, before being reconstructed and output in a good, valid hierarchically structured JPEG File.

The actual processes consists of several phases.

**Load Phase (Load File into Necessary Data Structures)**

This phase loads all of the marker data of the JPEG file, incuding its header data, loads its middle data (data which is not part of a marker or header) into a byte buffer, and loads all data (marker segment and middle data) into a byte buffer,

**Identification Phase (File Format Type Identification)**

1) The first pass consists of identification of the file type using an identification probability matrix. There are several flavors of the JPEG standards and the probability matrix determines the most likely flavor for the file being dealt with. Examples are JFIF, EXIF, etc. The comparisons are made by analyzing all APP and JPG codes, and other known marker data specific to that file format (JFIF or EXIF) type, cross referencing against probability data of known file types (the probability matrix). A user given probability factor can all for adjustment for the accuracy of these passes by adjusting constraints.

An example ID Probability Matrix:

Probabilities are based off lookup tables that can vary and be adjusted by the user.

Marker 1: Probability JFIF 10%, Probability EXIF 20%, Probability another file type 15%, etc

Marker 2: similar format to above

…

Marker n: similar format to above

Probabilities in the matrix above are determined by referencing file data for each marker in statistical look-up tables that area precomputed. The method of generating these look-up tables involves creating a database of probabilities based off of the input of a large number of different sample input JPEGS.

The question is, what factors should be “looked up” to get a good judge of the probabilities? An ANN could be used to generate the tables but may take a long time to code and train.

The dimensions of the probability matrix (x,y) combined with the probabilities as a z dimension can be used to generate a three dimensional surface that is scaled via one or several user input scalars (as discussed above). The final determined file format will be dependent upon the shape of this surface.

2) The second pass identifies the mode of operation (hierarchy, sequential, etc.) in a manner similar to the first pass, through the use of look-up tables/probabilitiy matrices.

3) The third pass identifies the method of encoding using a similar method as to the first two passes.

**Validation Phase**

1. The first pass ensures that data in the header file is valid for the file type that has been chosen.
2. The second pass ensures that data in the headers is valid for the mode of operation given above
3. The third pass ensures data in the headers is valid for the type of encoding.

**First Reconstruction Phase (Image Data Recovery)**

Data is analyzed sequentially through multiple passes to reconstruct the lost image.

A possible method to perform this operation is use of an image recovery function with parameters trained by an ANN. However, this would be very difficult and complex to implement.

**Trimming Phase (Junk Data Removal)**

The trimming phase passes through the file and deletes all invalid data that cannot be reconstructed or does not fit with the rest of the reconstruction.

**Second Reconstruction Phase (Header Reconstruction)**

All necessary headers for the JPEG file that are missing or are in valid are reconstructed and inserted via information known already about the file format. (i.e., SOI, EOI, SOS, and other such headers).

**Final Phase**

All necessary aggregate data is collected from data structures, combined and output as a valid JPEG file of the given type in the standard JPEG file format (using the JPEG file’s hierarchy). The file is output and possible made for display.

**Applications of Recovery of lost Matrix Data**

(Applied to image recovery)

Given an image defined by a matrix V of pixels, where each element vij = (r0, r1, r2), corresponding also to the vector uij, where each uij is also equal to a pixel in the matrix V.

Then there exists approximation functions Oμ(i,j, r0, r1, r2) that may be generated from the vectors uij through the process of Newtonian Approximation.

Lost, or corrupt, data can be removed prior to the generation of this function, and missing or corrupt data can be recognized where discontinuity(s) exist in this function.

The, these discontinuities can be recovered using **functional continuity recovery**, mentioned below.

**Functional Continuity Recovery**

Given an unknown continuous function r(x), which may be partitioned into three piece-wise functions,

F(x) = r(x) [a,b]

L(x) = r(x) [b, c]

G(x) = r(x) [c, d]

In this case, the outer bounds (a and d) may be infinite. That is, a may be equal to negative infinity, and d equal to infinity or vice versa for the pair if we so choose.

The function we have denoted as L(x) is to be our unknown function, that is, a piece of the function that we do not have sufficient data to represent.

Given the above, we can state that there exists an unknown function M(x) which “reasonably” approximates L(x). That is, M(x) approximates L(x) within a low value of ε. Where ε > 0.

Let there exist an operator known as an “interpolation” operator, Γ, and let there exist an “inverse interpolation” operator, Γ-1, and let there exist two functions:

A(x) [b,c]

B(x) [b,c]

Where

A(x) [b,c] = Γ-1 M(x)  
B(x) [b,c] = Γ-1 M(x)

And A(x) != B(x), A(x) > M(x) > B(x)

That is, the approximation function, M(x,) is sandwiched between A(x) and B(x).

A(x) and B(x) are given to have a certain probability of influencing each other in a chaotic system and are not independent of each other.

Through the process of using the interpolation operator, a series of n approximation functions can be interpolated to produce an accurate interpolation function μ(x), that closely represents L(x). The higher the degree of n, the more approximations interpolated, and the closer μ(x) approaches L(x).

M(x) = Γ M1(x) \* Γ M2(x)

M(x) =

By their nature, every equation follows a specific, inherent, pattern. To determine L(x) we must consider each independent vector of the equation r(x) and then “sum” (interpolate) them, retrieving an equation out of the interpolation of those vectors to represents F(x), G(x), and L(x), and therefore, from those, r(x). This process will find the missing segment of the curve L(x) by examining the patterns of the individual pieces (vector valued functions of…) the functions F(x) and G(x) by examining and incorporating their general behavior into L(x).

This process is logically equivalent to interpolation of a large number of splines in order to find its general approximation function. This iterative approximation is also similar in function to the rectangle rule used for integration, or the use of exhaustion to determine the area of a circle through polygons of increasing detail.

One possible solution is:

Where v(x) is a vector valued function for r(a) and ‘a’ is a point on the curve r(x). Each vector valued function vi(x) is treated as a Mi(x). The interpolation of these vector valued functions as approximations allows us to to generate the complete function, r(x).

(See Kolmogorov Complexity for how this may work as a corollary)

Given that v(x) is a vector describing the derivative at some point on the function r(x), and r(x) describes the joined partitioned functions F(x), L(x), and G(x), r(x) describes a very accurate close, approximation for the L(x) function, as well as its other component functions F(x) and G(x).