O objetivo da compressão de imagem não destrutiva é de representar um sinal de imagem com o menor número de bits possíveis sem perder informação, acelerando assim a transmissão e minimizando o espaço mínimo para armazenamento. Esta redução no bit rate é necessária para permitir muitas aplicações de realtime envolvendo a gestão e transmissão da informação audiovisual. Por exemplo uma imagem colorida não comprimida ou um frame de vídeo com uma resolução media de 500x500 pixels precisaria de 100 segundos de transmissão usando uma ISDN link tendo uma capacidade de 64000 bits por segundo (64 Kbps). O delay que resultaria disto é muito grande, considerando que um delay tão pequeno quanto 1 ou 2 segundos é necessário para conduzir um slide show interativo e um delay ainda mais pequeno (ordem dos 0.1 segundos) é necessário para transmissão de vídeo ou playback. A compressão não destrutiva é possível devido á redundância significativa presente nos sinais da imagens.

Na codificação não destrutiva a informação da imagem decodificada deve ser idêntica tanto quantitativamente (numericamente) e qualitativamente (visualmente) á imagem originalmente codificada. Esta necessidade preserva exatamente a veracidade da representação, também limita a quantidade de compressão que poderá ser atingida para um fator de compressão de dois ou três. Para atingir maiores fatores de compressão, os métodos de codificação não destrutivos tentam remover informação irrelevante. Este métodos necessitam que as imagens codificadas e decodificadas sejam apenas visualmente e não necessariamente numericamente idênticas.

Esquemas de codificação destrutiva fazem uso de componentes de codificação não destrutiva para minimizar a redundância no sinal que está a ser comprimido.

O codificador toma uma imagem como entrada e gera como saída um bitstream comprimido. O descodificador toma a bitstream comprimido como entrada e recupera a imagem original descomprimida. Em geral, o codificadora e o descodificador podem ser vistos como constituídos por três etapas principais.

1. Transformação: Esta fase aplica uma transformação reversível (um a um) aos dados de imagem de entrada. O objetivo desta fase é converter os dados de imagem de entrada em um formulário que pode ser comprimido de forma mais eficiente. Para o efeito, a transformação selecionada pode ajudar na redução da correlação de dados (interdependência, redundância), alterar a distribuição estatística dos dados e/ou embalar uma grande quantidade de informação em poucas amostras de dados ou regiões de sub-banda. As transformações típicas incluem mapeamento diferencial/preditivo, transformações unitárias como a transformação discreta do DCT, decomposições de sub-banda tais como transformações de wavelet, e conversões de espaço de cor, tais como conversão da representação RGB altamente correlacionada para a representação de luminância-crominagem menos correlacionada. Nesta fase, pode ser utilizada uma combinação das transformações acima. Por exemplo, uma imagem de cor RGB pode ser transformada para a sua representação luminância-crominagem seguida por DCT ou decomposição sub-banda seguida de mapeamento preditivo/diferencial. Em algumas aplicações (por exemplo, baixa potência), pode ser desejável operar diretamente nos dados originais sem incorrer no custo adicional da aplicação de uma transformação; neste caso, a transformação poderia ser definida para o mapeamento de identidade.

2. Mapeamento de dados para símbolo: Esta fase converte os dados de imagem em entidades chamadas símbolos que podem ser codificados de forma eficiente pela fase final. A conversão em símbolos pode ser feita através de divisórias e/ou codificação de comprimento de execução (RLC), por exemplo. Os dados de imagem podem ser divididos em blocos, agrupar amostras de dados vizinhas; neste caso, cada bloco de dados é um símbolo. Agrupar várias unidades de dados permite a exploração de qualquer correlação que possa estar presente entre os dados de imagem, podendo resultar em rácios de compressão mais elevados em detrimento do aumento da complexidade da codificação. Por outro lado, cada unidade de dados separada pode ser considerado um símbolo sem qualquer agrupamento ou partição. A ideia básica por trás do RLC é mapear uma sequência de números numa sequência de pares de símbolos (run, valor), onde o valor é o valor de uma amostra de dados na sequência de dados de entrada e o run é o número de vezes que a amostra de dados é repetida contíguamente. Neste caso, cada par (run, valor) é um símbolo. Implementações diferentes podem usar um formato ligeiramente diferente. Por exemplo, se a sequência de dados de entrada tiver longos períodos de zeros, alguns codificadores como a norma JPEG utilizam o valor para codificar apenas o valor das amostras de dados “não-zero” e executar para codificar o número de zeros anteriores a cada amostra de dados “não-zero”. O mapeamento adequado dos dados de entrada em símbolos é muito importante para otimizar a codificação. Por exemplo, o agrupamento de pontos de dados em pequenos conjuntos localizados, onde cada conjunto é codificado separadamente como um símbolo, permite que o esquema de codificação se adapte às características locais em mudança dos dados de imagem (transformados). O mapeamento dado-a-símbolo adequado depende da aplicação considerada e das limitações na complexidade de hardware/software.

3.Codificação de símbolos sem perdas: Esta fase gera um bitstream binário atribuindo palavras de código binárias aos símbolos de entrada. A codificação de símbolos sem perdas é geralmente referida como codificação silenciosa ou apenas codificação sem perdas, uma vez que esta fase é onde é realizada a codificação sem perdas real na corrente de bits comprimido final. As duas primeiras fases podem ser consideradas fases de pré-processamento para mapear os dados numa forma que pode ser codificada de forma mais eficiente por esta fase de codificação sem perdas. A compressão sem perdas é geralmente conseguida utilizando palavras-código de comprimento variável, onde as palavras de código mais curtas são atribuídas aos símbolos que ocorrem com mais frequência. Esta atribuição de código de comprimento variável é conhecida como codificação de comprimento variável (VLC) e também como codificação de entropia. Os codificadores de entropia, como Huffman e codificadores aritméticos, tentam minimizar a taxa média de bit (número médio de bits por símbolo) necessária para representar uma sequência de símbolos, com base na probabilidade de ocorrência de símbolos. Os códigos universais, tais como os códigos Elias e os códigos Exp-Golomb, são códigos de comprimento variável que podem codificar valores inteiros positivos em palavras de código binárias de comprimento variável sem conhecimento da verdadeira distribuição de probabilidades dos valores inteiros em ocorridas. Uma forma alternativa de obter a compressão é codificar cadeias de símbolos de comprimento variável usando palavras de código binárias de comprimento fixo. Esta é a estratégia básica por trás dos códigos do dicionário (Lempel-Ziv). O código sem perdas gerado (bitstream) deve ser unicamente descodificável; ou seja, o bitstream pode ser descodificado sem ambiguidade, resultando em apenas uma sequência única. Para códigos de comprimento variável, a decodificabilidade única é conseguida impondo uma condição de prefixo que diz que nenhuma palavra-código pode ser um prefixo de outra palavra-código. Os códigos que satisfazem a condição do prefixo são chamados códigos prefixos ou códigos instantaneamente descodificáveis, e incluem códigos Huffman, aritmética, Elias e Exp-Golomb. Os códigos prefixos binários podem ser representados como uma árvore binária, e também são chamados códigos estruturados por árvores. Para os códigos dicionários, a decodificabilidade única pode ser facilmente alcançada uma vez que as palavras de código geradas são de comprimento fixo.

NORMAS A TER EM CONSIDERAÇAO PARA ESCOLHER UM ALGORITMO

1. Eficiência da compressão: A eficiência da compressão é geralmente dada sob a forma de uma relação de compressão que compara o tamanho dos dados originais da imagem de entrada com o tamanho da bitstream compressa gerada. A eficiência da compressão também é geralmente expressa como uma taxa média de bit B em bits por pixel, ou bpp para abreviar. Para a codificação sem perdas, a eficiência de compressão alcançável é limitada pela entropia do conjunto finito de símbolos gerados como saída da fase 2, assumindo que estes símbolos são codificados separadamente, numa base um a um, pela fase 3.

2. Atraso de codificação: O atraso de codificação pode ser definido como o tempo mínimo necessário tanto para codificar e descodificar uma amostra de dados de entrada. O atraso de codificação aumenta com o número total de operações aritméticas necessárias. Também geralmente aumenta com um aumento nos requisitos de memória, uma vez que o uso da memória geralmente leva a atrasos de comunicação. Minimizar o atraso de codificação é especialmente importante para aplicações em tempo real.

3.Complexidade de implementação: A complexidade da implementação é medida em termos do número total de operações aritméticas necessárias e em termos dos requisitos de memória. Em alternativa, a complexidade da implementação pode ser medida em termos do número exigido de operações aritméticas por segundo e dos requisitos de memória para a obtenção de um determinado atraso de codificação ou desempenho em tempo real. Para aplicações que limitem o consumo de energia, a complexidade da implementação incluiria também uma medida do nível de consumo de energia. Uma maior eficiência de compressão pode geralmente ser alcançada aumentando a complexidade da implementação, o que, por sua vez, levaria a um aumento do atraso na codificação. Na prática, é desejável otimizar a eficiência da compressão, mantendo os requisitos de implementação o mais simples possível. Para algumas aplicações, como a navegação e recuperação de bases de dados, apenas é necessária uma complexidade de decodificação baixa, uma vez que a codificação não é realizada com tanta frequência como a decodificação.

4. Robustez: Para aplicações que requerem transmissão da bitstream comprimido em ambientes propensos a erros, a robustez do método de codificação para erros de transmissão torna-se uma consideração importante.

5. Escalabilidade: Os codificadores escaláveis geram um bitstream em camadas incorporando uma representação hierárquica dos dados de imagem de entrada. Desta forma, os dados de entrada podem ser recuperados em diferentes resoluções de forma hierárquica (escalabilidade em resolução), e a taxa de bits pode ser variada dependendo dos recursos disponíveis usando a mesma bitstream codificada (escalabilidade na taxa de bits; a codificação não tem de ser repetida para gerar as diferentes taxas de bits). A norma JPEG 2000 é um exemplo de um codificador de imagem escalável que gera um bitstream incorporado e que suporta a escalabilidade em componentes de qualidade, resolução, localização espacial e imagem.