**UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES  
FACULTAD DE CIENCIAS EXACTAS**

**INGENIERÍA DE SISTEMAS**



**Trabajo Práctico Especial   
2da Entrega**

**TEORÍA DE LA INFORMACIÓN**

**Integrantes:**

Sanchez Casco, Angelo : [angelo16s.as@gmail.com](mailto:angelo.16s@gmail.com), LU: 249116

Bertino, Ariel: [eugenioingenio10@gmail.com](mailto:eugenioingenio10@gmail.com), LU: 247865

# **Resumen**

El presente se enfoca en la codificación de la imagen entregada por la NASA (imagen de salida) mediante dos métodos distintos y decodificar la misma, por último se desea verificar el ruido del canal de comunicación a partir de otra imagen enviada a través de ese canal que funcionaria como imagen de llegada.

# **Introducción**

Contando con la imagen original (dividida en bloques) y a partir de tres valores de entropía arbitrarios se selecciona para cada bloque entre dos métodos de compresión el método de Huffman y RLC (Run length Coding), luego descomprimirla. Para finalizar de se desea analizar el ruido del canal de transmisión a partir de otro imagen dada que funciona como imagen enviada por el canal.

**Desarrollo**

Describiremos lo que se está calculando con explicaciones breves y mediante pseudocódigo, como se calculó.

Para codificar mediante Huffman:

Se debe generar el árbol con la codificaciones:

ordenar vecProb creciente

eliminar ceros(tonos inexistentes)

listaArboles =

para vecProb[i]

listaArboles.add(crearArbHuff(tono, prob))

mientras tamaño(listaArboles)>1

h=crearArbHuff(tonoInvalido, listaArboles[1]+listaArboles[2])

extraer(listaArboles[1] , listaArboles[2])

insertaOrdenado(listaArboles, h)

fin vector

retornar listaArboles[1]

codificación en sí:

codificacion=

cadena=

para cada pixelXY del bloque

c=pixelXY.extraerTono()

cadena.concatenar(codificar(c))

fin bloque

byte1=

para cada cadena(i)

si cadena(i) ==1 y byte.long < 8

byte1.concatenar(1)

si no si (byte.long < 8)

byte1.concatenar(0)

si no

codificacion.agregar(byte1)

byte1=

fin cadena

retornar codificacion.

Decodificación mediante huffman:

para obtener tonos, a partir de una codificación y la

cantSimbCodificados y el árbol de huffman de la

cabecera de la codificación:

tonos=

aux=arbol

b=codificacion.get()

mientras codificacion<>

si not(aux.esHoja())

si b.primero() == 1

aux=aux.getDer()

si no

aux=aux.getIzq()

SLL(b, 1)

si no

tonos.add(aux.getTono())

si b.long == 0

b = codificacion.get()

retornar tonos

para cada bloque de la reconstruccion:

tonos=generarTonos(codificacion)

para cada pixelXY de reconstruccion

pixelXY = tonos.obtenerTono()

retornar recontruccion

Para codificar mediante RLC, con tolerancia t(cero), con lo cual es RLC sin pérdida

codificacion=

cantColor=0

tonoAct=bloque[0][0].extraerTono()

para cada pixelXY del bloque

tonoSig=pixelXY.extraerTono()

si cantColor<255 y |tonoSig-tonoAct|<=t

cantColor++

si no

codificacion.agregar(tonoAct)

codificacion.agregar(cantColor)

cantcolor=0

tonoAct = TonoSig

fin bloque

retornar codificacion

Cuando decodificamos un bloque codificado con RLC hacemos:

a partir del ancho y alto del bloque obtenidos del bloque codificado

reconstruccion=

para cada pixelXY de la reconstruccion

codificacion.obtener(tono,cantidad)

mientras(cantidad > 0)

pixelXY.setTono(tono)

x++; y++; cantidad--

retornar reconstruccion

Para obtener el ruido del canal debemos construir la matriz condicional:

a partir de una imagen de partida y otra de llegada:

matrizConjunta=

vectorPro=0

para cada pixelXY de imagenSalida

cMS=pixelXY.getTono()

cLS=ImagenLlegada[X][Y].getTono()

vectorProb[cMS]++

matrizConjunta[cML][cMS]++

fin matrizConjunta

cada elemento(X,Y) de matrizConjunta

if vectorProb[Y] <> 0

matrizConjunta[X][Y]/=vectorProb[Y];

fin matrizConjunta

para concluir reutilizamos el método de Entropía Condicional así:

matCond = obtenerCanal(imagenSalida,imagenLlegada)

vecProb = obteneterVectorProb(ImagenSalida)

retornar = obtenerEntropiaCondicional(vecProb, matCond)

**Resultados**

A la hora del análisis de los resultados utilizamos diferentes valores de entropía umbral(ht) para decidir qué método de comparación utilizamos sobre cada bloque. Los valores que usamos son 3,2, 3,8 y 4,2.

los tamaños comprimidos son los siguientes 3.06 MB, 3,86 MB y 6,5 MB respectivamente. Como último parámetro de comparación calculamos el tasa de compresión que es un valor que funciona como la relación entre el tamaño original de la imagen y el tamaño comprimido. Estos valores de tasa de compresión son 1,55, 1,24 y 0.78.

Observamos que los bloque codificados con el método de Huffman pesan menos que los bloques codificados con el método RLC.

Para finalizar y como último parámetro calculado obtuvimos el ruido del canal que nos dice que dado un símbolo de llegada cuantas preguntas en promedio tenemos que hacer para dilucidar qué tono fue el de salida. El valor obtenido fue de 1,752 lo que nos estaría indicando que la longitud en promedia de la codificación de fuente de llegada respecto de la fuente de salida es de 1,752 bits. como dato adicional cabe destacar que es un valor coherente ya que cada tono se comprende un valor entre 0 y valores.

**Conclusión**

Como cierre al análisis de los datos se puede señalar que cuanto más grande sea el valor del umbral más bloques serán codificados con el método de RLC con lo cual el tamaño de la compresión será mayor. Luego es de suma relevancia saber que con un valor alto de umbral de entropía y al codificar muchos de los bloques con el método de RLC, tasa de compresión decae por debajo de uno, con lo cual el tamaño final de la compresión es mayor que el original, debido a que no hay corridas muy largas con tonos uniformes, con lo cual no conviene usar un umbral muy grande a la hora de seleccionar un metodo de compresion por sobre otro.

Lo que se puede concluir con respecto al ruido del canal y observando la imagen de llegada es que el canal de alguna manera no puede procesar los tonos altos del rango de grises con lo cual tiene tendencia a distorsionar los tonos claros, esto se puede ver en la en la imagen en zonas donde deberían predominar tonos claros se terminan encontrando ruido producto del canal.

**Aclaracion**: Nos vemos en la obligación de recalcar que después de un arduo trabajo de seguimiento y debugeo(incluso con profesores) del código fuente que realizamos no pudimos salvar el hecho de que a la hora de decodificar(reensamblar la imagen codificada) se nos distorsionan los bloque codificados con el método de Huffman. Aun así quisimos enviarles el progreso de la segunda entrega para que vean y evalúen nuestro trabajo.