Construcción de imágenes de súper-resulución usando un metodo probabilista

Pável Herrera Domínguez

Instituto Nacional de Astronomia Optica y Electronica

Abstract. La generación de imágenes de súper-resolución son imágenes generadas a partir de imágenes pequeñas, es uno de los problemas inversos mal planteados, el cual tiene varias aplicaciones en medicina, análisis de imágenes de satélite, entre otras. En este trabajo se implementa una de las alternativas para construir imágenes de súper-resolución usando Campos Aleatorios de Markov. Estas técnicas tienen ventajas contra otras técnicas, pues tienen la capacidad de manejar del ruido en la imagen original, el cual va implícito en el modelo.

Key words: reconstrucción,campos de markov,imágenes,súper-resolución

1 Introducción

Existen muchas razones por las que es necesario tener una forma de generar imágenes de súper resolución usando imágenes de baja resolución. Entre las razones tenemos por ejemplo imágenes de mejor calidad para un análisis el análisis de la imagen, extraer información de esta, resaltando la resolución de la imagen. Existen varias técnicas para hacer esto, entre los mas usados por los productos de software son interpolación lineal, cubica, usar filtros gaussianos. En [1] mencionan que existen dos divisiones principales para resolver o atacar este problema. Por un lado están los metodos que resuelven el problema en un dominio de la frecuencia. Por otro lado están los que resuelven el problema en un dominio espacial, entre estos destacan los metodos de interpolación, retropropagación iterativa, metodos estocásticos, entre otros En este trabajo se analizara un enfoque bayesiano el cual se clasifica entre los metodos estocásticos, en el cual la idea es modelar la imagen como un campo aleatorio de Markov, tomar la información de la imagen original como las observaciones, y el resto de la información faltante como información desconocida. Y estimar la información faltante como el conjunto de valores mas probables dadas las observaciones originales.

2 Trabajos relacionados

Este tema ha sido desarrollado varias veces en varios trabajos. En [1] y [2], hacen una revisión de los trabajos que existen relacionados con la generación de imágenes de súper resolución ya sea para vídeos, o para una imagen sola.

3 Metodología y Desarrollo

Los metodos de estimación bayesiana son muy buenos cuando se usan en problemas relacionados con conocimiento a priori. En este caso queremos encontrar los valores que maximizan una función de probabilidad dadas las observaciones que se tienen. Para ello usaremos los Campos Aleatorios de Markov.

3.1 Campos Aleatorios de Markov

Los campos aleatorios de Markov surgen como una extinción a las cadenas de Markov, en los cuales se sustituyen los índices temporales por índices espaciales. Mientras en las cadenas de Markov se tenían estructuras que dependían del solo del estado anterior $P(S_t|S_1...S_{t-1}) = P(S_t|S_{t-1})$ ahora tenemos estados dependen de los vecinos, digamos $P(s_x|\{s_y\}_{y\in V_x})$

Para definir un campo aleatorio de Markov necesitamos definir un conjunto de posiciones S, para cada $s \in S$ un conjunto de estados Γ_s, y un sistema de vecindades. Pasando esto al contexto de las imágenes tenemos que S es cada píxel,Γ_s son los valores que puede tomar el píxel correspondiente, por ejemplo en las imágenes en tonos de grises son los números enteros en el intervalo $[0, 255]$ si es una imagen a color $[0, 255]X[0, 255]X[0, 255]$, y finalmente el sistema de vecindades podemos tomarlo como el conjunto de píxeles que estén a una distancia D bajo alguna métrica como la distancia Mannhatan, sea menor a cierto valor que definido digamos 1, esto nos da los cuatro vecinos de cada píxel.

3.2 Campos de Gibbs

Decimos que un campo es de Gibbs si la distribución de probabilidad se puede escribir como

$$\Pi(x) = \frac{e^{-H(x)}}{Z} \tag{1}$$

donde $H(x)$ es una función de *energía* que induce el campo y Z es:

$$Z = \sum_{z\in\Omega} e^{-H(z)} \tag{2}$$

3.3 Reconstrucción de la imagen

Ahora que ya tenemos estas herramientas quisiéramos estimar cuales son los valores mas probables de cada uno de los píxeles al hacer la sśúper-escala, en los lugares que tenemos vacíos. En la siguiente ecuación X denota el vector de valores de toda la imagen, y_i denota los valores conocidos.

$$X = argmax_X(P(X|y_1, ..., y_P)) \tag{3}$$

Usando bayes tenemos que la solución al problema anterior es

$$X = argmax_X(\log P(y_1, ..., y_P|X) + \log P(X)) \tag{4}$$

Astutamente podemos definir P(x) y $P(y_1, ..., y_P|X)$ que es el conocimiento a priori de la imagen de súper-escala y el ruido estadístico de la información. Dado esto podemos decir que P(x) es descrito como un campo de Gibbs con una función de densidad dada por una función del la forma de la ecuación 1

Haciendo las cuentas necesarias tenemos que encontrar la imagen tal que maximiza la probabilidad anterior es equivalente a resolver:

$$X = argmin_X(\sum_{x \in I} U_c(x) + \lambda \sum_{y \in Y} F(x - y)) \tag{5}$$

Donde U_c es una función de energía que depende de los vecinos, y F es una función para medir la energía entre los

Para minimizar esta función es posible usar técnicas de decenso de gradiente o iteraciones de Guass-Seidel, esto se puede cuando F y U son funciones convexas.

3.4 Algoritmo

Algoritmo que reconstruye una imagen a cierta escala

```
Image reScaleImage (originalImage,scale,lambda)
begin
  Image sx,estimated,nextImage,originalScale;
  Initialize(sx,estimated,scale,originalScale)
  FOR iterations =1 to MAX_ITERATIONS
  begin
    FOREACH pixel in estimated
      C=sx[pixel]+lambda*numerOfNeightbours
      nextImage[pixel]=(sx[pixel]*originalScaled[pixel]+lambda*sumNeighbours(pixel))/C
    FOREACH pixel in estimated
      estimated[pixel]=nextImage[pixel]
  end
  return estimated
end.
```

Donde, en el código anterior **sx** representa los píxeles que se conoce su valor a priori es uno si su valor se conoce y cero si no. La imagen **originalScale** representa los valores o estados que se conocen a priori.

3.5 Detalles de la implementación

En la implementación en lugar de escalar directamente la imagen de una imagen pequeña a una imagen grande, se fue escalando gradualmente hasta llegar al tamaño deseado. Se uso una vecindad sencilla, un píxel es vecino de otro píxel si comparten un lado, visto de otra forma, los píxeles a distancia Manhatan 1 pertenecen a la vecindad. La implementacion se hizo en $C++$ usando OpenCV. La generación de ruido se hizo usando la función *add_white_noise* de Halcon.

4 Experimentos y resultados

Una de las ventagas de este tipo de formas de resolver el problema, es que es tolerante a cierto grado de ruido, pues el modelo asume que lo hay. Para experimentar con esto se eligieron un conjunto de imágenes de prueba las cuales ver figura 1, a las cuales miden 512x512 originalmente y se redujeron a imágenes 128x128 y 64x64, a estas imágenes ya reducidas se les agrego ruido gaussiano, generado artificialmente, la imagen resultante se calculo el error cuadrático medio contra la imagen original. Este error se comparo con el error cometido por la interpolación cubica de Gimp (software de linux para imágenes equivalente a Paint de windows).

Fig. 1. De izquierda a derecha truck, lena, wall

En la tabla 4 vemos los resultados al correr este algoritmo en las imágenes anteriores y compararlo con las imágenes generadas por GIMP.

Imagen	Tam.Original	Re-escalado	MSE-GIMP	MSE-CAM
lena	64	512	266.623	1006.155
lena	128	512	110.148	711.213
wall	64	512	233.632	1208.582
wall	128	512	103.788	642.819
truck	64	512	840.288	1017.763
truck	128	512	387.601	496.087

Table 1. La tabla muestra el error cuadrático medio de las imágenes generadas por un algoritmo CAM con un factor de suavizamiento de $\lambda = .01$

En las figuras 2 y 2 se muestra la imagen de lena generada por un algoritmo que usa CAM, y las imágenes generadas por GIMP.

Luego experimente con imágenes con ruido blanco aditivo, generado artificialmente en Halcon, solo se le agrego ruido a las imágenes submuestreadas. Dandonos los resultados de la tabla 4. Las imágenes a súper-escalar se pueden

Fig. 2. A la izquierda la imagen generada con CAM y a la derecha una interpolación cubica generada por GIMP. El tamaño original era 128 y fueron aumentadas a 512

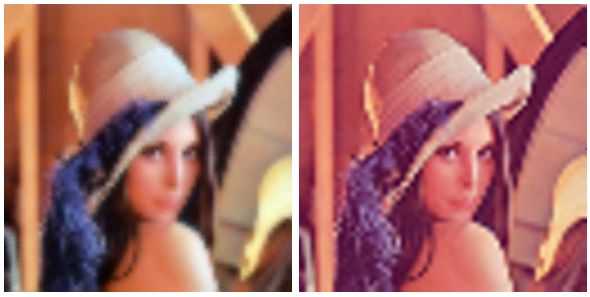

Fig. 3. A la izquierda la imagen generada con CAM y a la derecha una interpolación cubica generada por GIMP. El tamaño original era 64 y fueron aumentadas a 512

se pueden ver en las figuras 4 y 5, las cuales claramente fueron escalas por el editor de este documento.

Fig. 4. A la izquierda la imagen de lena y a la derecha la imagen wall, ambas con ruido aditivo, esta imagen mide 128 x 128

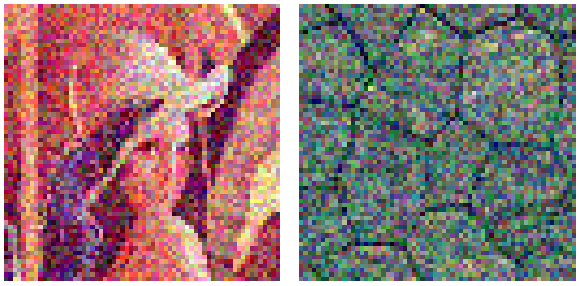

Fig. 5. A la izquierda la imagen de lena y a la derecha la imagen wall, ambas con ruido aditivo, esta imagen mide 64 x 64

En las figuras 6 y 7 se muestra la imagen de lena generada por un algoritmo que usa CAM, y las imágenes generadas por GIMP pero con ruido blanco aditivo. Mientras que en las figuras 9 y 8 se muestran las imágenes generadas para la imagen wall.

4.1 Analisis

Los resultados comparando solo el MSE pareciera que las imágenes son de menos calidad, aunque cualitativamente podemos notar que las imágenes generadas en por el algoritmo basado en CAM's preserva los bordes mejor, aunque los *difumina* mas por el parametro *lambda*. Cabe mencionar también que el error cuadrático medio también es mayor pues la escala que tiene en RGB las imágenes finales no necesariamente terminan con el mismo rango dinámico que la imagen original.

Imagen	Tam.Original	Re-escalado	MSE-GIMP	MSE-CAM
lena-noise	64	512	990.430	741.783
lena-noise	128	512	850.239	466.139
wall-noise	64	512	1002.244	1016.491
wall-noise	128	512	867.896	564.850

Table 2. La tabla muestra el error cuadrático medio de las imágenes generadas por un algoritmo CAM con un factor de suavizamiento de $\lambda = .01$, las imágenes sub-muestreadas tienen ruido blanco aditivo

Fig. 6. A la izquierda la imagen generada con CAM y a la derecha una interpolación cubica generada por GIMP. El tamaño original era 128 y fueron aumentadas a 512

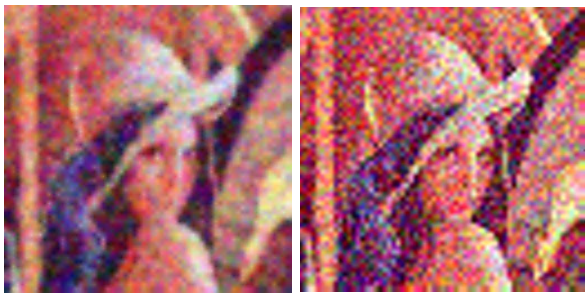

Fig. 7. A la izquierda la imagen generada con CAM y a la derecha una interpolación cubica generada por GIMP. El tamaño original era 64 y fueron aumentadas a 512

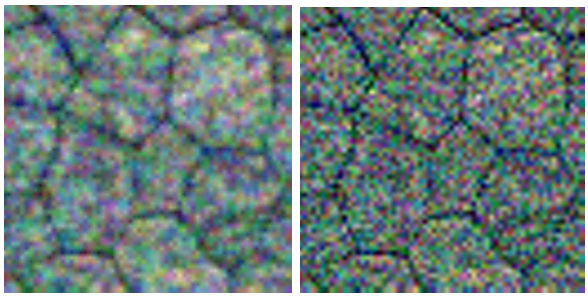

Fig. 8. A la izquierda la imagen generada con CAM y a la derecha una interpolación cubica generada por GIMP. El tamaño original era 64 y fueron aumentadas a 512

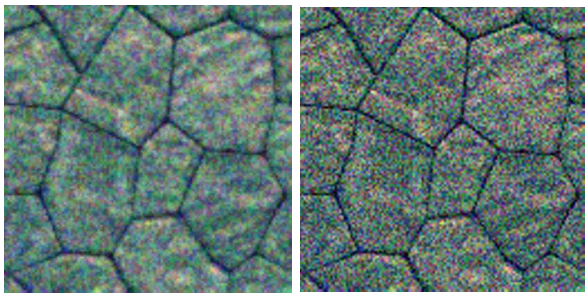

Fig. 9. A la izquierda la imagen generada con CAM y a la derecha una interpolación cubica generada por GIMP. El tamaño original era 128 y fueron aumentadas a 512

Aunque hice un intento de corregir este fenómeno, esto no se arreglo del todo. En las imágenes se puede ver este fenómeno, pues las imágenes generadas por el CAM son mas claras.

5 Conclusiones y trabajo futuro

Como conclusiones podemos decir que este tipo de técnicas aunque son lentas, tienen beneficios bastante notorios. Claramente tienen un mejor funcionamiento cuando las imágenes tienen ruido es algo que hay que remarcar.

En cuanto a trabajos futuros, seria bueno probar cosas que aunque ya se han intentado como hacer una estimación inicial con interpolación, y luego re-estimarla usando CAMs. Por otro lado buscar formas de que sea mas rápido pues tiene que hacer varias iteraciones para converger.

References

1. Robert L. Stevenson Sean Borman. Super-resolution from image sequences - a review.
2. Moon Gi Kang Sung Cheol Park, Min Kyu Park. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, 2003.