Ajuste Robusto de Lineas según la Distancia Ortogonal

Luis Gerardo de la Fraga

Departamento de Computación, Cinvestav, Av. IPN 2508, 07360 México, D.F.
Correo-e: fraga@cs.cinvestav.mx

Resumen

Es este trabajo se ajusta de forma robusta líneas a un conjunto de puntos dados, minimizando la suma de las distancias ortogonales de los puntos a la línea. Este es un problema no lineal que se resuelve con el algoritmo evolutivo llamada evolución diferencial. El algoritmo resultante se aplica para extraer de forma robusta todas las líneas presentes en una imagen. Se muestran resultados de simulaciones y con imágenes reales.

1. Introducción

El problema de ajuste de un conjunto de puntos a una línea tiene una solución lineal si se considera la suma del cuadrado de la distancia en la ordenada del punto dado al modelo de la línea. Esta es una solución muy bien conocida del ajuste por mínimos cuadrados de un conjunto de puntos a una línea. Y como el problema es lineal, se tiene una expresión algebraica para calcular a y b del modelo de la línea $y = a + bx$.

Las alternativas que se tienen al ajuste son la la transformada de Hough [1] y por el método de momentos [2].

La desventaja de usar mínimos cuadrados es que si existen puntos que no pertenecen a la recta, conocidos como *puntos distantes* (en inglés, *outliers*), entonces la línea ajustada se verá grandemente afectada debido a que el ajuste se verá influenciado por el cuadrado de la distancia de ese punto distante a la recta, lo que produce que la recta no pase sobre el conjunto de puntos dado.

Este trabajo es una primera aproximación al ajuste robusto usando la suma de los valores absolutos de las distancias Euclidianas, y su aplicación para extraer de forma robusta las líneas presentes en una imagen. La función valor absoluto no tiene una derivada continua en su mínimo por lo que no se puede resolver algebraicamente el problema. Aquí se resuelve este ajuste por medio de una heurística llamada evolución diferencial. Al usar el valor absoluto, en vez del cuadrado de la distancias, la influencia de los puntos distantes se reduce y no afecta la localización de la línea ajustada, como se demostrará más adelante.

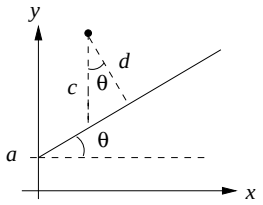

Figura 1: Esquema para deducir la expresión para la distancia perpendicular de un punto a la línea.

En la Sec. 2 se presenta el problema de ajuste y como se resuelve usando la evolución diferencial. En la Sec. 3 se muestran resultados de extracción de líneas en imágenes reales. Finalmente en la Sec. 4 se presentan las conclusiones y trabajo a futuro que podría realizarse.

2. Ajuste según la suma de las distancias ortogonales

Dado un conjunto de n pares de coordenadas (x_i, y_i), para $i \in [1, n]$, se quiere ajustar una línea $y = a + bx$ tal que se minimice la suma de la distancia perpendicular de cada punto dado (al menos dos) a la línea, esto es minimizar:

$$\frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - (a + bx_i)|}{\sqrt{1 + b^2}} \tag{1}$$

La expresión de la distancia del punto a la línea puede deducirse fácilmente: de la Fig. 1, a es el punto de cruce de la línea con el eje de las ordenadas, b es la pendiente de la línea y es igual a $\tan(\theta)$, c es la distancia vertical del punto a la línea, $c = |y_i - (a + bx_i)|$, y $\cos(\theta) = d/c$, por lo que $d = c\cos(\theta)$ y expresando $\cos(\theta)$ según la $\tan(\theta)$ se llega a la expresión de (1).

El problema de minimización expresado por (1) fue resuelto con el algoritmo evolutivo llamado evolución diferencial [3, 4], que será explicado brevemente en la subsección 2.2. Al ser un algoritmo evolutivo, la ED

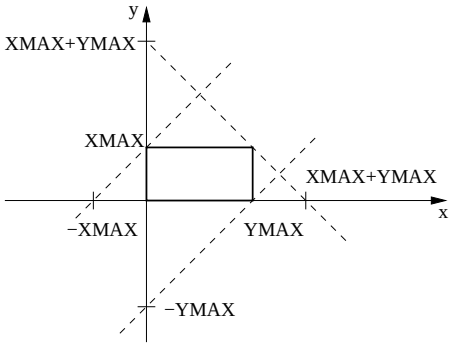

Figura 2: Esquema para encontrar el dominio para la variable a.

propone un conjunto de soluciones iniciales aleatorias para el problema. Estas soluciones se combinan según los operadores de la ED, se seleccionan las mejores soluciones y después de varias interacciones de combinación y selección se obtiene la solución al problema expresado en (1). Esta es una descripción muy somera de como trabaja la ED.

La ED puede verse como un algoritmo de búsqueda dirigida, por lo que se necesita encontrar el dominio para las variables involucradas. Para b, en vez de usarla directamente, se puede usar $b = \tan(\theta)$, y usar el dominio para $\theta \in [0, 180]$. Para la variable a, si se tiene una imagen como se esquematiza con un rectángulo en la Fig. 2, entonces cuando $0 \leq |b| < 1$ se tienen el dominio para $a \in [-\text{YMAX}, \text{XMAX}+\text{YMAX}]$. Si $|b| \geq 1$, entonces resulta mejor numéricamente usar el modelo $x = (y - a)/b$, y el dominio para a (que ahora representa el punto de cruce de la línea ajustada con el eje de las abscisas) es entonces $[-\text{XMAX}, \text{XMAX} + \text{YMAX}]$. En general se puede usar el dominio $a \in [\text{mín}(-\text{XMAX}, -\text{YMAX}), \text{XMAX}+\text{YMAX}]$ para una imagen rectangular.

2.1. Extracción de líneas

Algorithm 1 Algoritmo para extraer m líneas de un conjunto de puntos P.

Require: El conjunto de puntos, P; el valor de m; el total de puntos n_T

Ensure: m líneas en el conjunto P

1: **for** $i = 1 : m$ **do**
2: linea$_i$ = evolucion_diferencial(P)
3: n = borra_puntos(P, linea)
4: $n_T \leftarrow n_T - n$

Aquí el problema es minimizar (1) y maximizar al mismo tiempo el número de puntos que se usan en el ajuste. Para que el problema sea de un solo objetivo, entonces se minimiza

$$g = -\frac{10n}{n_T} + \sum |d_i| \tag{2}$$

donde n es el número de puntos que pertenecen a la línea y n_T el número de puntos totales; la distancia d_i es la distancia perpendicular de la línea al punto i e $i \in [1 : n]$. Para contar los puntos de la línea solo se consideran los puntos a una distancia menor a un valor U.

El algoritmo que encontraría m líneas en una imagen se muestra en el Algoritmo 1.

2.2. Evolución diferencial

La evolución diferencial es un algoritmo evolutivo que trabaja en poblaciones de individuos y en el que sobreviven los individuos más aptos. La población que usa la ED está compuesta por un conjunto de individuos, o vectores de números reales. Todos los vectores se inicializan con números aleatorios con una distribución uniforme dentro de los límites de búsqueda para cada variable.

Existen varias versiones de la ED. Aquí se usó la versión rand/1/bin porque es robusta y provee los mejores resultados para diferentes pruebas y problemas de optimización [5].

Actualmente existe gran cantidad de literatura sobre la ED; el lector podría referirse a [4] como un buen punto de inicio para conocer los detalles de la ED.

El pseudocódigo de la ED se muestra en el Algoritmo 2. La base de la ED está en el ciclo de las líneas 8-12: un nuevo individuo se genera a partir de tres individuos diferentes escogidos aleatoriamente; cada valor del nuevo vector (que representa al nuevo individuo) se calcula del primer padre más la diferencia de los otros dos padres multiplicada por F, la constante de diferencia; el nuevo vector se calcula si un número aleatorio (entre 0 y 1) es menor que R, la constante de recombinación de la ED. Para prevenir el caso de que un nuevo individuo sea igual al primer padre, al menos un elemento del vector es forzado a calcularse a partir de los otros padres, esto está en la línea 9 del pseudocódigo, cuando $i = i_{\text{rand}}$, y i_{rand} es un entero aleatorio entre 1 y n. Cuando el nuevo individuo se evalúa, si éste es mejor que el padre (en las lineas 11-12), entonces reemplaza al padre. La condición de paro usada aquí es: si el número de iteraciones es menor que 10,000, o cuando la diferencia en las valores de la función objetivo del peor y mejor individuo es menor que 0.001. Esta condición de paro se llama criterio *diff* en [6], y es la que se recomienda para una tarea de optimización global.

Algorithm 2 Algoritmo de la evolución diferencial versión rand/1/bin

Require: El domino de la búsqueda; el valor de s de la condición de paro. Los valores para el tamaño de la población, μ; el número máximo de generaciones, M; las constantes de deferencia y recombinación, F y R, respectivamente.

1: initializar($X = \{\mathbf{x}_1, \ldots, \mathbf{x}_\mu\}$)
2: evaluar(X)
3: $k = 0$
4: **repeat**
5: **for** $j = 1 : \mu$ **do**
6: Sean r_1, r_2 y r_3 tres enteros aleatorios en $[1, \mu]$, tal que $r_1 \neq r_2 \neq r_3$
7: Sea i_{rand} un entero aleatorio en $[1, n]$
8: **for** $i = 1 : n$ **do**
9: $$x'_{i,j} = \begin{cases} x_{i,r_3} + F(x_{i,r1} - x_{i,r2}) & \text{si} \\ \quad U(0,1) < R \text{ ó } i = i_{rand} \\ x_{i,j} \text{ de otro modo} \end{cases}$$
10: $x'_{n+1,j} = \text{evaluar}(\mathbf{x}'_j)$
11: **if** $x'_{n+1,j} < x_{n+1,j}$ **then**
12: $\mathbf{x}_j = \mathbf{x}'_j$
13: min $= x_{n+1,1}$, max $= x_{n+1,1}$
14: **for** $i = 2 : n$ **do**
15: **if** $x_{n+1,i} <$ min **then**
16: min $= x_{n+1,i}$
17: **if** $x_{n+1,i} >$ max **then**
18: max $= x_{n+1,i}$
19: $k \leftarrow k + 1$
20: **until** (max − min) $< s$ ó $k > M$

De acuerdo a las pruebas realizadas en la conferencia CEC 2005 [7], la ED es la segunda mejor heurística para resolver problemas de optimización global con parámetros reales, cuando el número de parámetros es alrededor de 10. La mejor heurística es una Estrategia Evolutiva llamada G-CMA-ES [8]. Aquí se usa la ED porque tiene un mejor tiempo de ejecución y porque es muy fácil de realizar.

3. Resultados

Se realizaron tres pruebas para el Algoritmo 1. En la Fig. 3 vemos el ajuste de una linea a 11 puntos, dos de los cuales pueden considerarse distantes. Todos los puntos fueron considerados en el ajuste. Como se ve en la Fig. 3, el ajuste que se calcula con la ED según la suma de las distancias absolutas no se ve afectado por la presencia de los puntos distantes. El dominio de búsqueda para la ED fue: $a \in [-10, 20]$, $b \in [0, 180]$, y

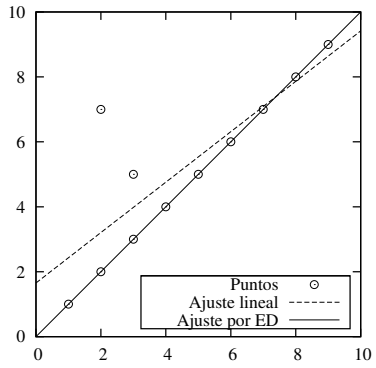

Figura 3: Ajuste lineal y ajuste según la suma del absoluto de las distancias.

se usaron los siguiente valores para la ED: tamaño de población $\mu = 30$, $s = 0.001$, $M = 10,000$, $R = 0.9$, $F = 0.7$.

En la Fig. 4 se extraen dos líneas de 471 puntos. Se realizaron tres ejecuciones con 10 %, 30 % y 60 % de puntos distantes adicionales. Claramente se ve que se pueden extraer correctamente las dos líneas. Aquí el espacio de búsqueda fue $a \in [-350, 700]$ y todos los demás parámetros igual a la primera prueba descrita en el párrafo anterior. Para esta prueba se usó $U = 10$.

Finalmente se hizo una prueba con la imagen real, de tamaño 1024×768 pixels, mostrada en la Fig. 5 a la izquierda. Está imagen fue cambiada a tonos de gris, sus bordes fueron extraídos; finalmente fue binarizada y todos los puntos extraídos. La Fig. 5 a la derecha se muestras las 20 líneas extraídas del conjunto de puntos. Aquí solo se cambio el espacio de búsqueda para a a $[-1000, 2000]$.

4. Conclusiones

Se presenta un algoritmo que extrae de forma robusta líneas de una imagen. El algoritmo se basa en minimizar la suma de las distancias absolutas y maximizar el número de puntos que pertenecen a la línea. Este es un problema de optimización no-lineal y además no tratable analíticamente (la función valor absoluto no tiene derivadas continuas en todo el espacio) fue resuelto con la heurística llamada evolución diferencial.

El algoritmo es muy sencillo y con las pruebas mostradas se verifica que funciona correctamente.

Como trabajo a futuro hace falta comparar con otros métodos robustos de ajuste, para conocer a fondo las ventajas y desventajas de usar heurísticas para resolver problemas de ajuste no lineales.

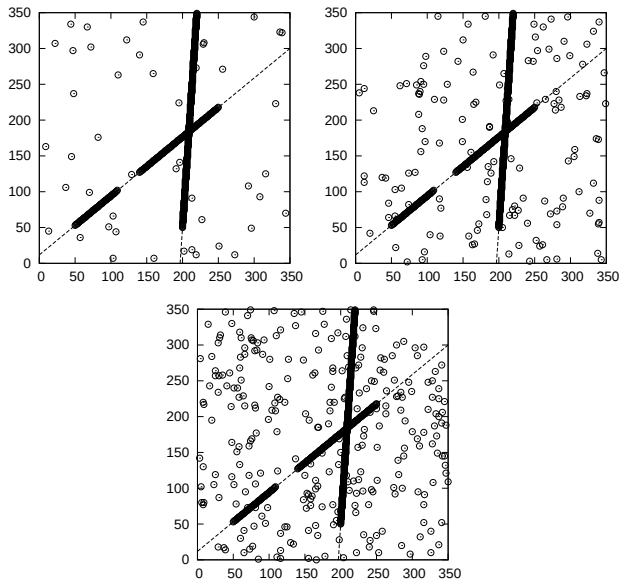

Figura 4: Búsqueda de líneas con 10 %, 30 % y 50 % de puntos distantes.

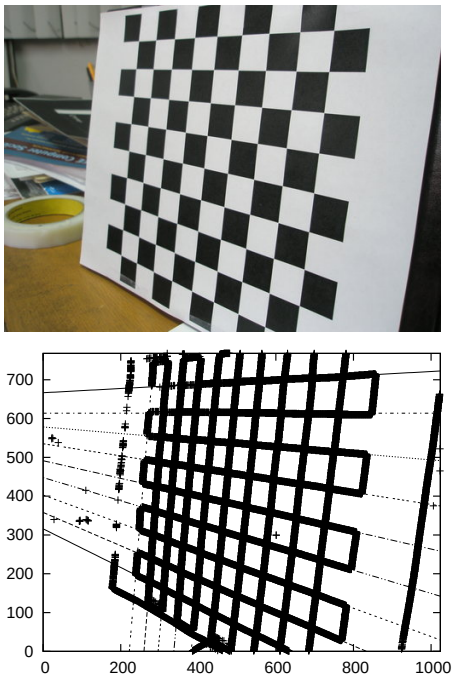

Figura 5: A la izquierda, la imagen real usado en la tercera prueba. A la derecha, los puntos extraídos y las 20 líneas extraídas.

Agradecimientos: Este trabajo fue apoyado parcialmente por el proyecto SEP-CONACyT 80965.

Referencias

[1] N. Kiryati and A.M. Bruckstein. Heterocedastic hough transform (htht): An efficient method for robust line fitting in the 'errors in the variables' problem. *Computer Vision and Image Undertanding*, 78:69–83, 2000.

[2] H. Qjidaa and L. Radouane. Robust line fitting in a noisy image by the method of moments. *IEEE Trans on PAMI*, 21(11):1216–1223, 1999.

[3] R. Storn and K.V. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuos spaces. *J. of Global Optimization*, 11(4):341–359, 1997.

[4] U.K. Chakraborty. *Advances in Differential Evolution. Studies in Computational Intelligence*. Springer, 2008.

[5] E. Mezura, J. Velázquez, and C. A. Coello. A comparative study of differential evolution variants for global opti mization. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 485–492, New York, NY, USA, 2006. ACM Press.

[6] K. Zielinski and R. Laur. Stopping criteria for differential evolution in constrained single-objective optimization. In *Advances in Differential Evolution*. Springer, 2008.

[7] N. Hansen. Comparisons results among the accepted papers to the special session on real-parameter optimization at cec-05, 2006. http://www.ntu.edu-.sg/home/epnsugan/index_files/CEC-05/com pare-results.pdf.

[8] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *Proceedings of the Congress on Evolutionary Computation (CEC-2005)*, pages 1769–1776, 2005.