

**UNIVERSIDAD COMPLUTENSE DE MADRID**  
**FACULTAD DE CIENCIAS FÍSICAS**

DEPARTAMENTO DE FÍSICA DE MATERIALES



**TRABAJO DE FIN DE GRADO**

Código de TFG: FM17

Técnicas avanzadas de microscopía aplicadas al estudio  
de nuevos materiales

Advanced microscopy techniques applied to cuttingedge  
material systems

Supervisor/es: María Varela del Arco

**Jesús Bautista Villar**

---

Grado en Física  
Curso académico 2021-2022  
Convocatoria Junio

[Título extendido del TFG (si procede)]

## Resumen:

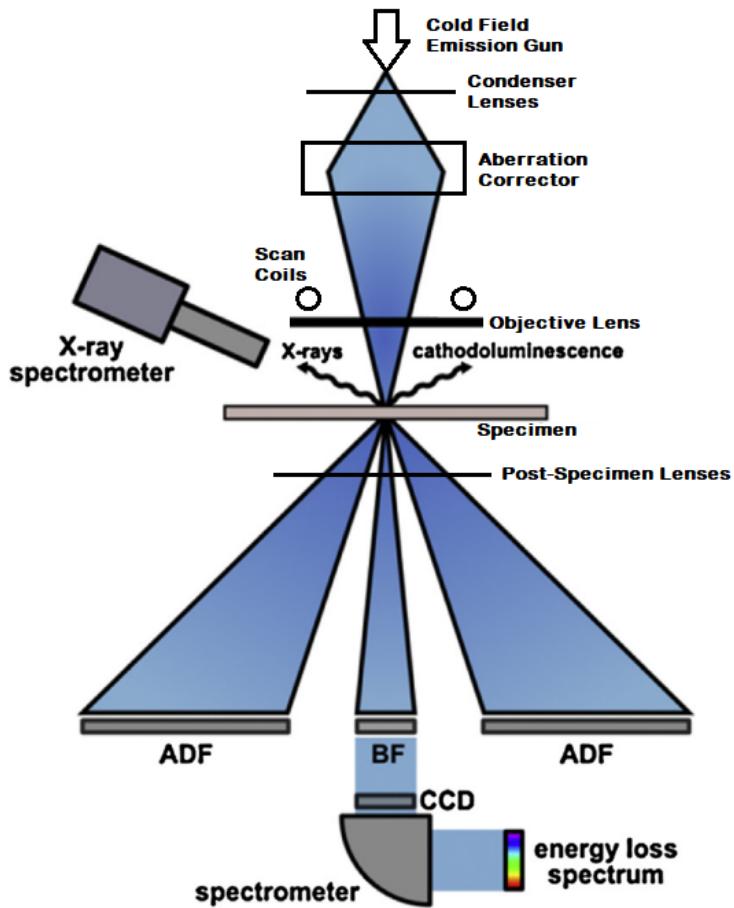
### Abstract:

# Índice

<b>Introducción</b>	<b>2</b>
<b>Objetivos</b>	<b>4</b>
<b>1. Tratamientos de datos multidimensionales</b>	<b>5</b>
1.1. Compresión y <i>denoising</i> de señales bidimensionales . . . . .	5
1.1.1. <i>Fast Fourier transform</i> (FFT) . . . . .	5
1.1.2. <i>Singular value decomposition</i> (SVD) . . . . .	6
1.2. Detección de columnas atómicas . . . . .	8
<b>2. Caracterización de columnas atómicas</b>	<b>11</b>
2.1. Métodos de factorización matricial . . . . .	12
2.1.1. <i>Principal component analysis</i> (PCA) . . . . .	12
2.1.2. <i>Non-negative matrix factorization</i> (NMF) . . . . .	13
<b>3. Clasificación de columnas atómicas e identificación de estructuras por IA</b>	<b>14</b>
3.1. Métodos para análisis de <i>clusters</i> . . . . .	14
3.1.1. <i>Support Vector Machine</i> (SVM) . . . . .	15
3.1.2. <i>K-means</i> . . . . .	15
3.2. <i>Deep Learning</i> (DL) . . . . .	15
<b>4. Búsqueda de vacantes de oxígeno en STO mediante imágenes 4D-STEM</b>	<b>17</b>
4.1. Procesamiento de imágenes ABF de STO . . . . .	18
4.1.1. Generación de parches con información sobre la celda unidad . . . . .	18
4.1.2. <i>Denoising</i> por PCA y NMF . . . . .	18
4.1.3. <i>Denoising</i> por una red neuronal con topología de <i>autoencoder</i> . . . . .	19
<b>Referencias</b>	<b>20</b>

# Introducción

Dentro del campo de la microscopía, la STEM (*Scanning Transmission Electron Microscopy*) es única en lo que respecta a la cantidad de información que se puede obtener de una única muestra. Es más, las capacidades de este tipo de microscopios han mejorado drásticamente en los últimos años con la aparición de los correctores de aberración multipolares, permitiéndonos alcanzar resoluciones que llegan a superar los 0.5 Å. Por ello hablamos hoy en día de esta técnica como una herramienta fundamental para el diseño de nuevos materiales avanzados.

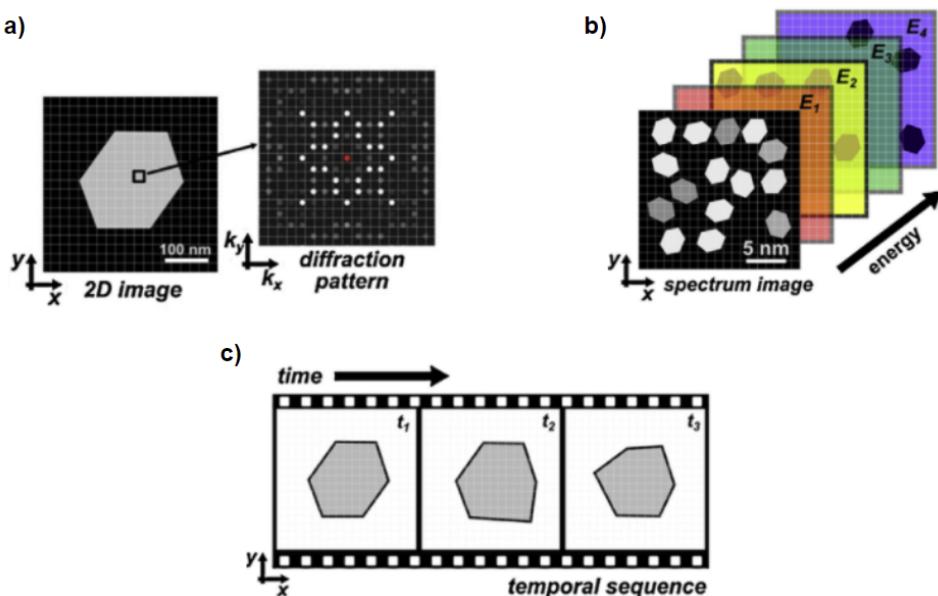


**Figura 1:** Esquema que muestra los componentes principales del STEM, adaptado de la referencia [1] añadiendo información suplementaria sobre ciertos instrumentos [5].

En la Figura 1 se muestra el esquema de un STEM de alta resolución. El haz de electrones es generado aplicando una diferencia de potencial de decenas o centenas de kV sobre una fina punta que históricamente siempre ha estado compuesta por tungsteno, aunque actualmente se han empezado a investigar nuevos cátodos de grafeno-níquel [2]. Dichos electrones logran superar la barrera de potencial por efecto túnel y posteriormente atraviesan una serie de lentes capaces de corregir el haz aplicando campos magnéticos (bipolares o multipolares) que pueden llegar al orden de varios teslas.

Una vez el haz de electrones alcanza la muestra, se pueden producir dispersiones tanto elásticas (curvatura de la trayectoria) como inelásticas (pérdida de energía cinética). Es entonces cuando entran en juego los detectores, que comúnmente podrán suministrarnos diversas señales:

- Los detectores HA (*High Angle*) medirán el **ADF** (*Annular Dark-Field*). Contarán en cada píxel la cantidad de electrones transmitidos que, tras atravesar la muestra, se han desviado mucho respecto al eje normal al plano de incidencia. Aquí juegan un papel protagonista los núcleos pesados, pues son capaces de interactuar fuertemente con los electrones.
- Para ángulos más pequeños tenemos el detector de **BF** (*Bright-Field*), que mediante mínimos en el patrón de difracción es capaz de localizar las columnas atómicas menos pesadas.
- Cuando los electrones se dispersan de forma inelástica pierden una cantidad de energía que puede ser medida por **EELS** (*Electron Energy Loss Spectroscopy*) mediante una CCD (*Charge-Coupled Device*) o una matriz de fotodiodos.
- Cubriendo la entrada al espectrómetro se puede situar una CCD removible capaz de medir el **patrón de difracción** completo que se genera en cada punto.
- Por separado, un espectrómetro es capaz de detectar las **emisiones de rayos X** generadas por las excitaciones de electrones en la muestra.



**Figura 2:** En esta figura se muestra esquemáticamente todas las dimensiones en las que se puede trabajar con STEM: a) Espacio real-recíproco con ADF y difracción, b) energía mediante EELS y/o rayos X, c) temporal si se requiere observar la evolución de un fenómeno. Imagen adaptada de la referencia [1] realizando una selección de imágenes.

Por esta razón, tal y como se muestra en la [Figura 2](#), el STEM puede aportar en una única medida datos en el dominio temporal que pueden vivir simultáneamente dentro del espacio real, recíproco y de energías. Definitivamente, contamos una gran cantidad de información que, debido a su alta dimensionalidad, plantea un escenario en el que resulta necesaria la introducción de nuevos métodos estadísticos avanzados de big data y algoritmos punteros de inteligencia artificial (IA).

En este trabajo nos centraremos principalmente en el 4D-STEM, donde encontraremos imágenes ADF y patrones de difracción. Ambos grupos de datos en conjunto serán los que nos aporten la información necesaria para identificar y cuantificar algunos de los defectos presentes en una red cristalina.

## Objetivos

- Estudiar la relevancia de la inteligencia artificial y algunas herramientas estadísticas para el tratamiento de datos dentro del campo de la microscopía electrónica.
- Obtener información física relevante de un sistema en base a una serie de datos multidimensionales difícilmente interpretables.
- Diseñar una red neuronal capaz de clasificar las columnas atómicas presentes en imágenes ADF.
- Construir una red neuronal que ayude a detectar defectos puntuales reduciendo el ruido en imágenes 4D-STEM.
- Localizar las columnas atómicas y etiquetarlas en función de su información característica.
- Manipular los datos experimentales mediante métodos estadísticos diseñados para reducir la dimensionalidad de un sistema.
- Explorar herramientas de compresión para optimizar el tamaño en memoria de las imágenes tomadas por STEM.

# 1. Tratamientos de datos multidimensionales

## 1.1. Compresión y *denoising* de señales bidimensionales

Uno de los grandes problemas ligados al manejo de datos multidimensionales es su tamaño en memoria. En experimentos que requieran el trabajo simultáneo de todos los sensores, la cantidad de bytes necesaria para guardar la información de una única muestra puede llegar al orden de varios gigabytes. Por ello se hace tan tedioso trabajar con este tipo de datos, y necesitamos métodos de compresión que nos ayuden en nuestras tareas de análisis. También hablaremos de *denoising* (filtrado del ruido) porque estos mismos algoritmos van a dar muy buenos resultados.

### 1.1.1. *Fast Fourier transform (FFT)*

Atendiendo a la definición de la transformada de Fourier, toda función periódica puede ser descrita como una base ortogonal de senos o cosenos. Si nos vamos al caso discreto, dada una sucesión de  $n$  puntos tendremos:

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-i2\pi jk/n} \quad f_k = \frac{1}{n} \left( \sum_{j=0}^{n-1} \hat{f}_j e^{-i2\pi jk/n} \right) \quad (1)$$

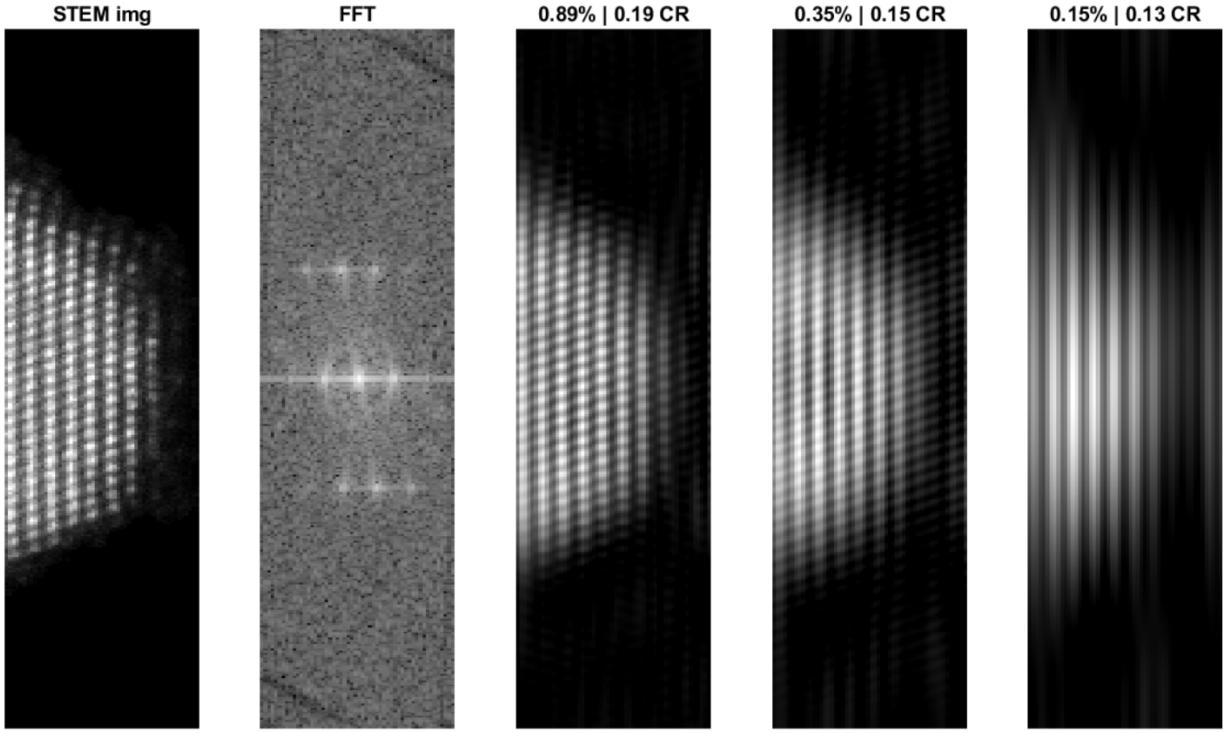
El algoritmo FFT nos permite calcular los coeficientes  $f_j$  con un ínfimo coste computacional. De este modo, podremos codificar en el espacio recíproco gran cantidad de información en cuestión de milisegundos.

La clave de trasladar nuestra imagen al espacio recíproco está en imponer un umbral al valor absoluto de los coeficientes  $f_j$ . Así somos capaces de seleccionar las frecuencias más significativas, que serán las que almacenemos en memoria como una matriz de números complejos tipo *float* de 4 bytes y valores lógicos de 1 bit para aquellas componentes nulas. De este modo lograremos reducir considerablemente el espacio en memoria ocupado por la información de nuestra imagen. Una vez que queramos reconstruir la imagen comprimida no tendremos más que cargar dicha matriz y aplicar la FFT inversa.

Para cuantificar esta compresión, se define el coeficiente de ratio de compresión  $CR$  como una relación proporcional entre el tamaño de la imagen original  $S_o$  y el de la imagen ya comprimida  $S_c$ :

$$S_c = CR \cdot S_o \quad (2)$$

En el caso de la [Figura 3](#), tenemos una imagen original con profundidad de 2 bytes (*uint16*). Al aplicar el algoritmo explicado previamente y calcular el ratio de compresión, observamos una reducción en el tamaño de más de cinco veces respecto al tamaño original. Estos resultados son bastante impresionantes, más aún si tenemos en cuenta que además de aplicar una compresión logramos reducir el ruido de forma considerable, pues estamos capturando la periodicidad de la red (ver el caso [3a](#)).



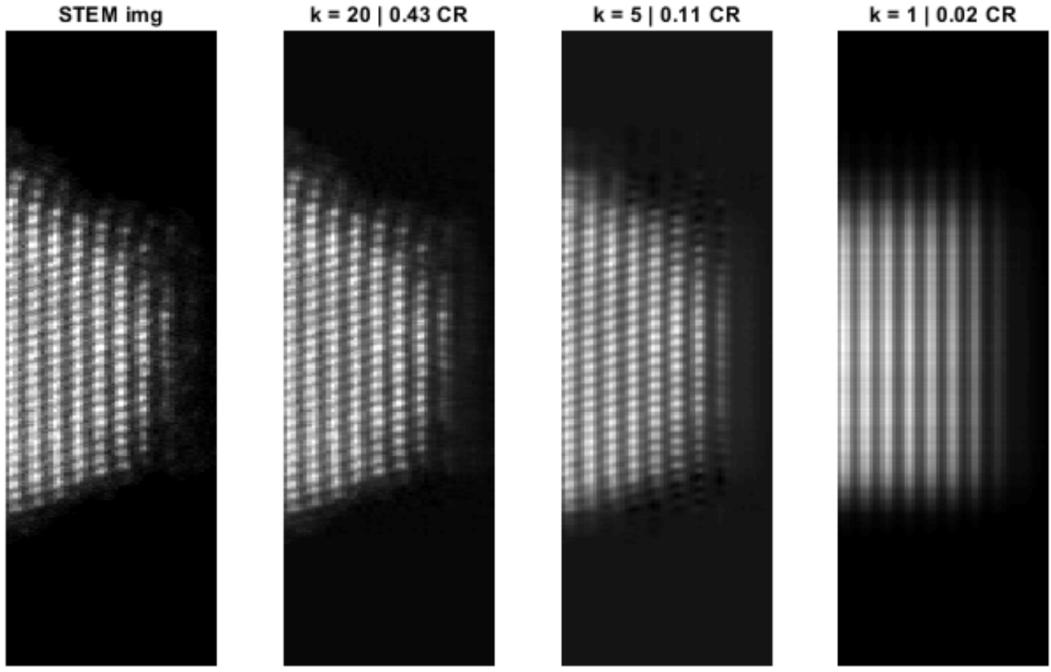
**Figura 3:** Dada una imagen STEM ADF de tungsteno (111) [7], tomamos la FFT para visualizar su información en el espacio recíproco. Posteriormente se aplican una serie de umbrales que eliminan los coeficientes menos significativos de la transformada, para cada caso se indica en el título el porcentaje de coeficientes no nulos respecto al total y el ratio de compresión. El código ha sido desarrollado en MATLAB por el autor y se puede encontrar en [13].

Aunque únicamente hemos puesto como ejemplo una imagen ADF, este método de compresión y limpiado de ruido funciona de forma similar para BF, patrones de difracción en el espacio recíproco e imágenes que representen una integración de todos los canales de energía en cada punto (EELS *mean image*).

### 1.1.2. *Singular value decomposition (SVD)*

Dada una matriz  $A$  con dimensiones  $m \times n$ , sus valores singulares corresponderán con la raíz cuadrada de los autovalores de  $A^T A$ , que vendrán denotados por  $\sigma_1, \dots, \sigma_n$  y ordenados de forma que  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$ . Con esto en mente, se puede demostrar [12] que existe una descomposición tal que  $A = U\Sigma V^T$ , donde  $U$  ( $m \times m$ ) y  $V$  ( $n \times n$ ) son matrices ortogonales y  $\Sigma$  ( $m \times n$ ) es una matriz diagonal compuesta por  $\sigma_1, \dots, \sigma_n$ .

Interpretaremos  $A$  como una transformación que nos lleva del dominio  $\mathbb{R}^n$  hasta el rango  $\mathbb{R}^m$ . Como es de esperar, ambos espacios podrán ser descritos por una base vectorial, que vendrá contenida en las matrices  $V$  (dominio) y  $U$  (rango). Con esto en mente, ya somos capaces de visualizar la SVD como una método que se asegura de encontrar un par de bases ortonormales tales que la transformación quede representada por la matriz diagonal  $\Sigma$ .



**Figura 4:** Sobre la misma imagen STEM de la [Figura 3](#) se ha realizado una descomposición SVD. En la figura se muestra la imagen que resulta de tomar los  $k$  valores singulares indicados, junto al ratio de compresión. Programa desarrollado en MATLAB por el autor [\[13\]](#).

Para construir la matriz  $V = [v_1 \dots v_n]$ , se debe encontrar una base ortonormal  $\{v_1, \dots, v_n\}$  compuesta por autovectores de la matriz simétrica  $A^T A$ . Estos mismos elementos  $v_i$  se aprovechan para diseñar la matriz  $U = [u_1 \dots u_m]$  tomando los vectores  $Av_1, \dots, Av_n$ , que también serán ortogonales (demostración sencilla en [\[12\]](#)).

Bajo esta construcción, resulta que  $\|Av_i\| = \sqrt{\lambda_i}$ , de modo que cada elemento  $Av_i$  se asocia a un valor singular de  $A$ . El problema de esto es que únicamente existen  $r < m$   $\sigma_r \neq 0$ , por lo que inicialmente no contaremos con la cantidad suficiente de vectores  $Av_i$  para formar una base de  $\mathbb{R}^m$ , tendremos que prolongar el conjunto  $\{u_1 \dots u_r\}$  de dichos vectores normalizados. Teniendo todo esto en mente, finalmente obtendremos la base  $\{u_1 \dots u_m\}$ , donde  $u_i = \frac{Av_i}{\|Av_i\|} = \frac{Av_i}{\sigma_i}$ .

Una vez llegados a este punto, ya somos capaces de desarrollar  $A = U\Sigma V^T$  para llegar a una de las propiedades fundamentales de esta descomposición (*low rank approximation property*):

$$\begin{aligned} A &= U \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix} V^T = U \left( \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} + \cdots + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix} \right) V^T \\ &= \sigma_1 u_1 v_1^T + \cdots + \sigma_r u_r v_r^T, \end{aligned}$$

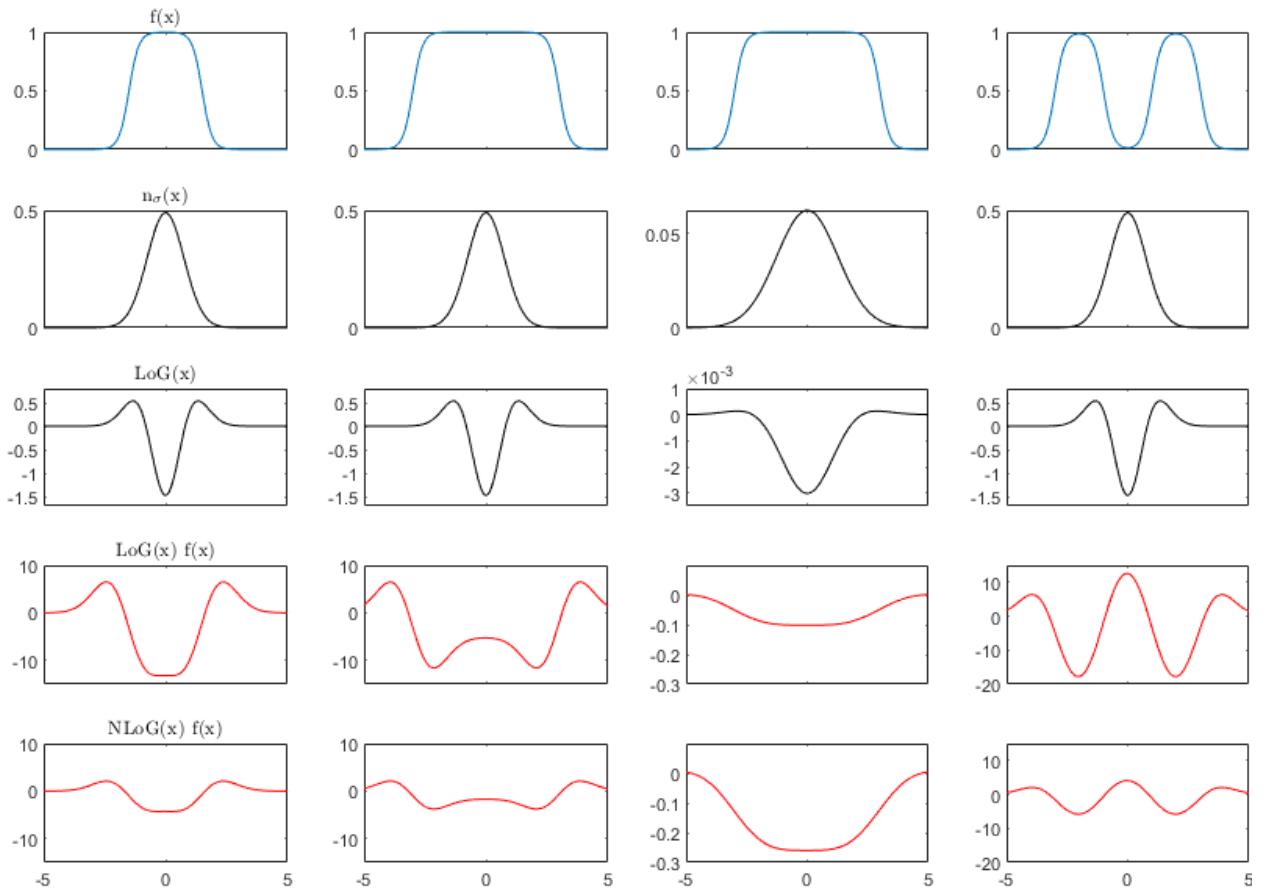
es decir, dada una matriz  $A$  con dimensiones  $m \times n$  y rango  $r$  siempre se cumplirá

$$A = \sum_{i=1}^r \sigma_i u_i v_i^T \tag{3}$$

Aprovechando esta propiedad podremos obtener una  $A_{ap}$  similar a la matriz  $A$ , mediante una aproximación en mínimos cuadrados que coincidirá con los  $k$  primeros términos de la ecuación (3). Cabe destacar que el rango de  $A_{ap}$  será dicho  $k \leq r$ .

## 1.2. Detección de columnas atómicas

Automatizar la detección de máximos y mínimos en nuestras imágenes es un paso fundamental. Necesitamos codificar las posiciones de las distintas columnas atómicas para posteriormente ser capaces de extraer la información relevante del sistema. Para ello recurriremos a distintas técnicas de visión por ordenador que nos permitan detectar picos de intensidad.



**Figura 5:** Imagen generada en MATLAB por el autor [13] que muestra el caso de un *blob* unidimensionales sobre los que se convoluciona distintos filtros NLoG. En la primera columna tenemos un *blob* estrecho que puede ser capturado por una gaussiana de bajo  $\sigma$ . En la segunda y tercera columna contamos con un *blob* más largo que requiere de una gaussiana con mayor  $\sigma$  para que la señal convolucionada cuente con un pico bien definido. Finalmente, la última columna muestra la convolución de una señal con dos *blobs*.

Aunque existen numerosos métodos, nosotros nos centraremos en los *scale-space methods* por ser los más usados en STEM. Tal y como se observa en la Figura 5, estos algoritmos toman inicialmente una señal matemática (filtro) y la convolucionan con la imagen real. En este caso trataremos el filtro NLoG (*Normalized Laplacian of Gaussian*). No obstante, también

suelen usarse la DoG (*Difference of Gaussians*), el HoG (*Hessian of Gaussian*) y la CHT (*Circle Hough Transform*, ver [Figura 6](#)). La magia de aplicar estas funciones reside en la forma de la señal convolucionada, pues si tomamos su valor absoluto presentará máximos en aquellos puntos donde la intensidad de la señal imagen es mayor. Comúnmente, la forma de dicha convolución dependerá del escalado del filtro, que en el caso del NLoG podremos controlar con un parámetro que llamaremos  $\sigma$ , como se muestra en la [Ecuación 4](#).

$$NLoG(x, y, \sigma) = \sigma^2 \nabla^2 n_\sigma = -\frac{1}{\pi \sigma^2} \left[ 1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (4)$$

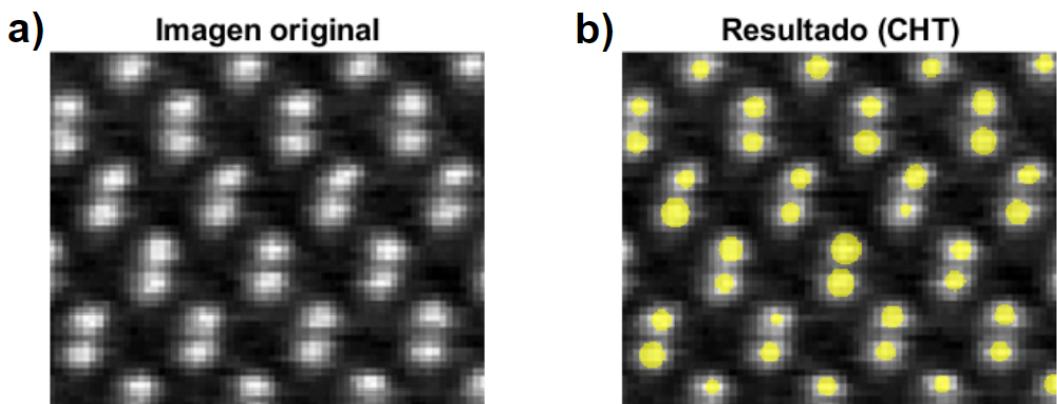
Estos máximos de la señal final no siempre se encontrarán bien definidos, dependerá del tamaño del pico de intensidad en la imagen y de la forma de la gaussiana. Es más, existe una relación proporcional entre  $\sigma$  para el que aparece el máximo y el tamaño del punto brillante, que podremos aprovechar para filtrar el ruido y evitar la detección de falsas columnas atómicas.

Al aplicar esta señal con varios valores de  $\sigma$  generamos un volumen de imágenes apiladas:

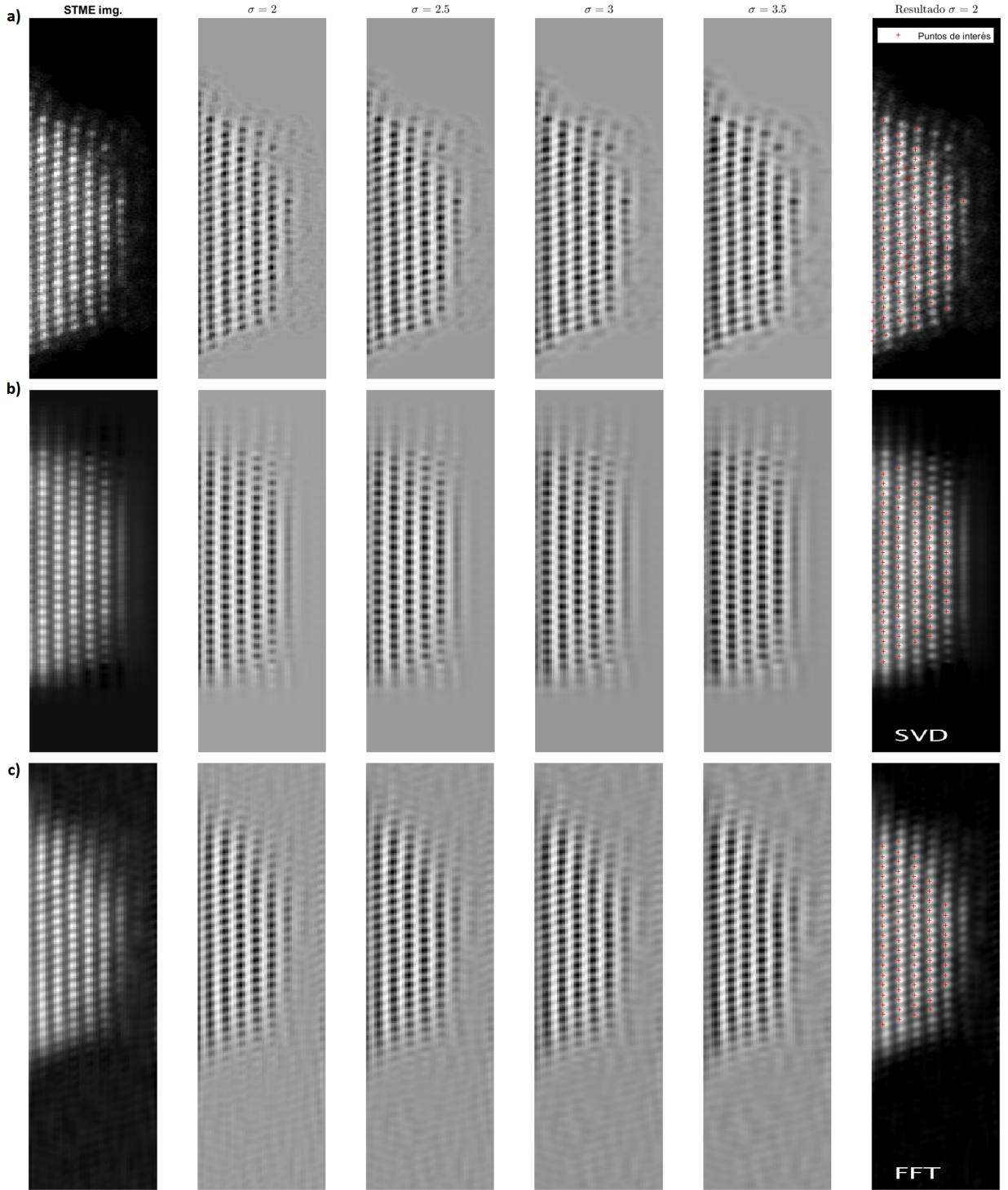
$$S(x, y, \sigma) = \sigma^2 \nabla^2 n_\sigma \cdot I(x, y) \quad (5)$$

De este modo, dada una imagen  $I(x, y)$ , podremos encontrar el conjunto de puntos brillantes  $(x^*, y^*)$  de tamaño  $\sigma^*$  sin más que buscar los máximos dentro de dicho volumen S:

$$(x^*, y^*, \sigma^*) = \arg \max_{(x, y, \sigma)} |\sigma^2 \nabla^2 n_\sigma \cdot I(x, y)| \quad (6)$$



**Figura 6:** Sobre una imagen ADF de GaAs [5] se utiliza la funcionalidad *Circle Finder* de la aplicación *Image Segmenter* incluida en el paquete *Image Processing and Computer Vision* de MATLAB, que aplica la CHT [10] como filtro para encontrar círculos de cierto radio.



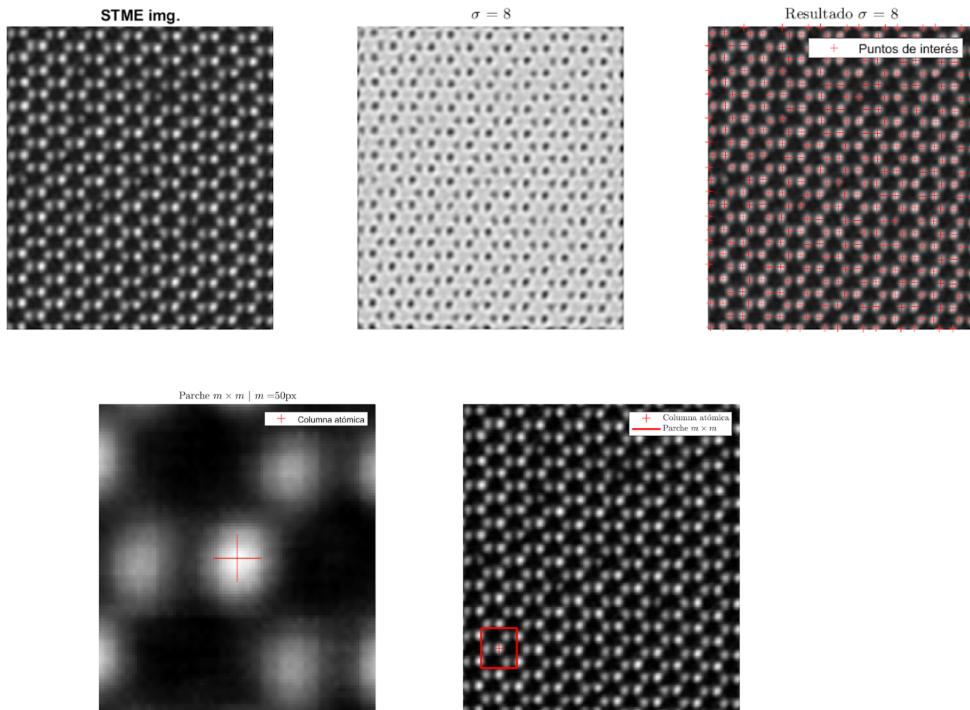
**Figura 7:** Tomando la misma imagen ADF de la [Figura 3](#). En todos los casos se muestra la imagen de entrada junto a los planos del volumen  $S$  que han sido generados aplicando el filtro NLoG para los valores indicados de  $\sigma$ . Una vez el algoritmo ha encontrado los blobs para  $\sigma = 2$ , se expone el resultado obtenido. En a) tenemos la imagen original, en b) y c) se ha aplicado SVD y FFT por separado para reducir el ruido. Programa desarrollado en MATLAB por el autor [13].

En la [Figura 7](#) se muestra un ejemplo de este algoritmo programado en MATLAB por el autor. Como se puede observar, los resultados son satisfactorios siempre y cuando no nos encontremos cerca de la frontera de la red. También se aplica el algoritmo de detección a una imagen filtrada por SVD para mostrar que podemos obtener mejores resultados si tratamos la entrada reduciendo el ruido y saturando los puntos brillantes.

## 2. Caracterización de columnas atómicas

Una vez conocemos la posición de cada columna atómica, necesitamos generar y asignarles una firma que nos permita clasificar las distintas especies de columnas en función del tipo de átomo que representan. Dichas firmas, mejor conocidas como vectores de características (*feature vectors*), serán el input que suministremos a una red neuronal y/o algoritmos de *clustering* para clasificar las distintas columnas atómicas presentes en cierta imagen ADF.

En nuestro caso, la información necesaria para caracterizar a cada columna la extraemos fijándonos en cómo se posicionan sus vecinos más próximos. Para cada átomo tomamos de la imagen original un pequeño parche  $m \times m$  (ver [Figura 8](#)) y lo almacenamos en un tensor  $m \times m \times n$ , donde  $n$  corresponderá con el número total de parches. Posteriormente, aplanamos cada sección  $m \times m$  para generar vectores de longitud  $m^2$ , de modo que finalmente nos quedaremos con una matriz  $X$  de dimensiones  $n \times m^2$ . En las siguientes secciones trataremos de procesar toda esta información para generar la firma de cada una de nuestras columnas atómicas.

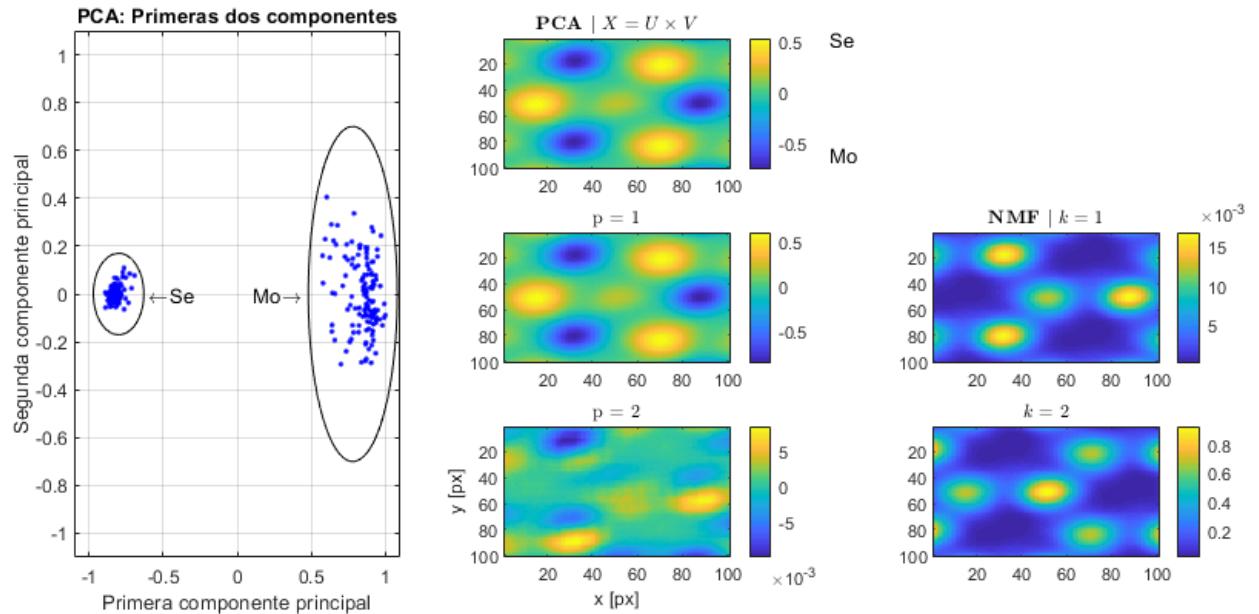


**Figura 8:** Para imagen ADF de MoSe<sub>2</sub>, adaptada de la referencia [11], se muestra el resultado del algoritmo de detección de columnas atómicas y la escala de los parches que se han seleccionado para caracterizarlas. Programa desarrollado en MATLAB por el autor [13].

## 2.1. Métodos de factorización matricial

Al estar codificando nuestros datos en forma matricial, podemos plantear la caracterización de las columnas como un problema de reducción de dimensionalidad. Trataremos a las  $n$  filas como distintas observaciones y a las  $m^2$  columnas como las distintas variables que se pueden medir en cada observación.

La idea será descomponer la matriz  $X$  en las matrices  $V$  ( $p \times m^2$ ) y  $U$  ( $n \times p$ ), de modo que  $X = U \times V$ . La dimensión  $p$  coincidirá con el número de componentes que contendrá el vector de características de cada columna atómica. Estos métodos se aplican principalmente en reducción de dimensionalidad porque típicamente  $p \ll m^2$ .



**Figura 9:** Para la misma imagen ADF de MoSe<sub>2</sub> que aparece en la [Figura 8](#), se muestran los resultados tras aplicar PCA, quedándonos con únicamente las dos primeras componentes, y NMF para un máximo de  $k = 2$ . Programa desarrollado en MATLAB por el autor [13], reproduce los resultados expuestos en [11].

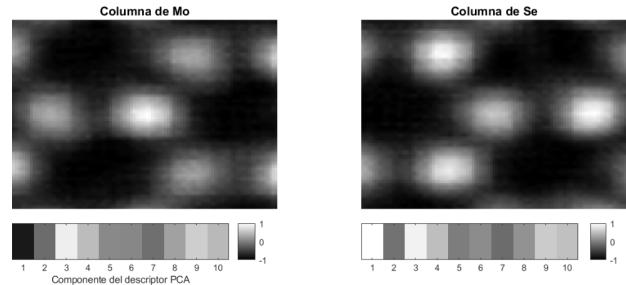
### 2.1.1. Principal component analysis (PCA)

El PCA es una técnica que, haciendo uso de la descomposición SVD, nos ofrece un sistema de coordenadas que toma como base las direcciones de máxima variación. Esto resulta muy útil para *datasets* que cuentan con una gran correlación entre algunas de sus entradas, pues seremos capaces de reducir drásticamente la cantidad de variables necesarias para distinguir entre sus distintas clases de observaciones. Para computar este algoritmo inicialmente se genera la matriz  $B$  substrayendo la media de la filas de  $X$ :

$$B = X - \bar{X} \quad \text{donde} \quad \bar{X} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \bar{x} \quad y \quad \bar{x}_j = \frac{1}{n} \sum_{i=1}^n X_{ij}. \quad (7)$$

Una vez tenemos  $B$ , ya todo se reduce a aplicar el método SVD. Los autovectores asociados a los valores singulares más grandes corresponderán con las direcciones de mayor variación.

En las Figuras 9 y 10 se muestran algunos de los resultados obtenidos por PCA. Observamos que, aunque logramos separar perfectamente los dos tipos de átomos, al contar con coordenadas negativas, es complicado encontrar una conexión entre las componentes principales y la física de las columnas atómicas

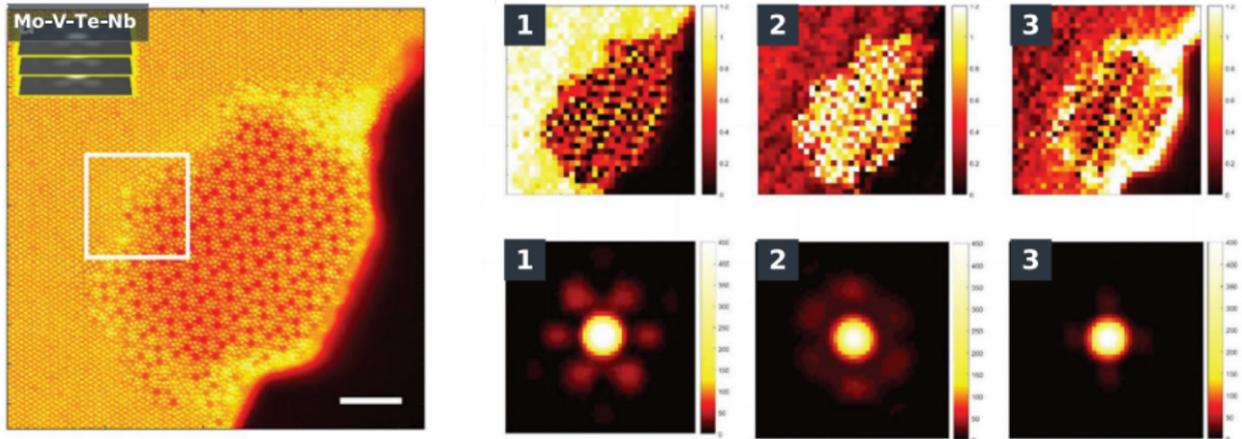


**Figura 10:** En función de los resultados obtenidos en la Figura 9 con PCA, se toman dos columnas atómicas arbitrarias (una de Mo y otra de Se) y se determinan sus vectores descriptores resultantes tras seleccionar  $p = 10$  componentes principales. Programa desarrollado en MATLAB por el autor [13].

### 2.1.2. Non-negative matrix factorization (NMF)

La NMF soluciona el problema de interpretabilidad del PCA, aportando una representación sin coordenadas negativas que respeta las restricciones físicas. Este hecho permite a la NMF realizar **mapas de fases** sobre imágenes ADF, como los que se ven en la Figura 11.

En el caso de la Figura 9 se realiza una NMF para dos componentes, lo que nos aporta una imagen promedio para cada tipo de parche.



**Figura 11:** Figura adaptada de la referencia [11]. Tenemos una imagen ADF de Mo-V-Te-Nb sobre la que se ha aplicado NMF de tres componentes. De seguido encontramos los mapas de fase ( $X = U \times V$ ) y su autovector asociado (reescalado de las filas de  $V$ ).

### 3. Clasificación de columnas atómicas e identificación de estructuras por IA

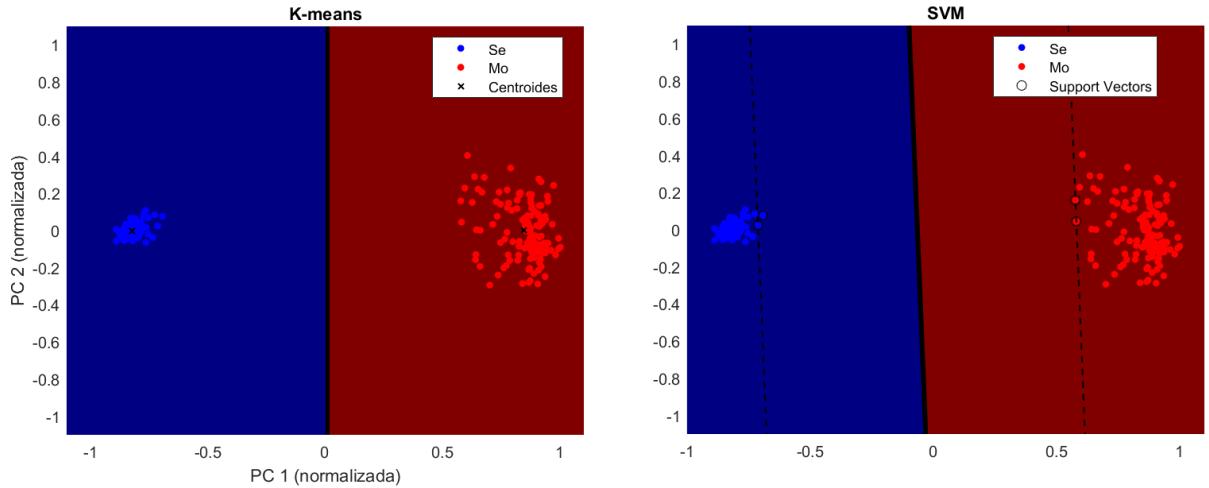
La filosofía de los algoritmos de ML (Machine Learning) es que los programas sean capaces de realizar tareas para las que no han sido programados. En nuestro caso, nos interesa aprovechar las grandes capacidades que estos modelos presentan para resolver problemas masivos de identificación de estructuras.

Una vez hemos logrado localizar y caracterizar nuestras columnas atómicas, ya podemos proceder con su clasificación. Para ello, como generalmente contaremos con una firma en más de 3 dimensiones, recurriremos a métodos basados en inteligencia artificial. En todos ellos, nuestra entrada siempre será la firma de una columna atómica, mientras que la salida corresponderá con su clase.

Para entrenar los modelos que exponemos en este trabajo (excepto *K-means*), nos basaremos en el **aprendizaje supervisado**, de modo que todos los datos de entrenamiento serán previamente analizados y clasificados. No obstante, hay que tener en cuenta que también es posible desarrollar algoritmos basados en aprendizaje no supervisado, semi-supervisado o reforzado.

#### 3.1. Métodos para análisis de *clusters*

El análisis de *clusters* es un método estadístico de procesamiento de datos. Su función principal consiste en organizar los distintos elementos de un *dataset* en grupos, teniendo en cuenta su distancia dentro del espacio que generan las variables de entrada.



**Figura 12:** Clasificación realizada por *K-means* ( $k = 2$ ) y SVM para el caso de la Figura 9. Se muestra la recta que separa las distintas regiones, los centroides de *K-means* y el margen entre clases del SVM. Para ajustar estos modelos se han aprovechado las funciones *fitcsvm* y *kmeans* contenidas en el paquete *Statistics and Machine Learning Toolbox* de MATLAB [13].

### 3.1.1. Support Vector Machine (SVM)

SVM es un algoritmo de aprendizaje supervisado que permite encontrar el hiperplano que mejor separa dos *clusters* de datos distintos. Tal y como se observa en la [Figura 12](#), el método se encarga de maximizar el margen entre clases, que corresponde con la anchura máxima de la región paralela al hiperplano que no contiene puntos de datos.

### 3.1.2. K-means

Este algoritmo parte de una serie de observaciones y se encarga de organizarlas en los  $k$  grupos que nosotros le indiquemos. Para llevar a cabo esta clasificación sigue un criterio muy simple: Asignar cada medida al *cluster* cuyo valor medio asociado sea el más cercano.

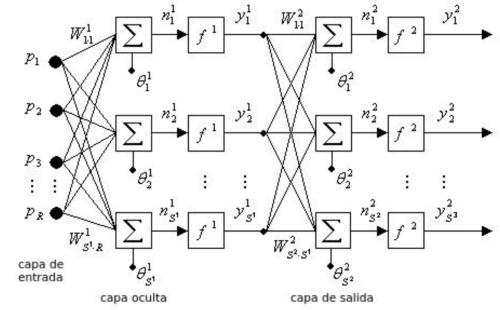
Dado que este método es de aprendizaje no supervisado, en vista de los resultados de la [Figura 12](#), si entre nuestros datos contáramos con algún tipo de vacante o intersticial, podríamos ajustar  $k$  para que dicho tipo de columnas atómicas pertenecieran a otro grupo distinto.

## 3.2. Deep Learning (DL)

El DL está contenido por definición dentro de los algoritmos de ML, no obstante, en este caso se hace uso de estructuras más complejas que tratan de simular el cerebro humano, las redes neuronales. Las técnicas de DL que han ganado más protagonismo en el campo de la visión por ordenador son las CNN (*Convolutional Neural Networks*), las RNN (*Recurrent Neural Networks*) y los transformadores.

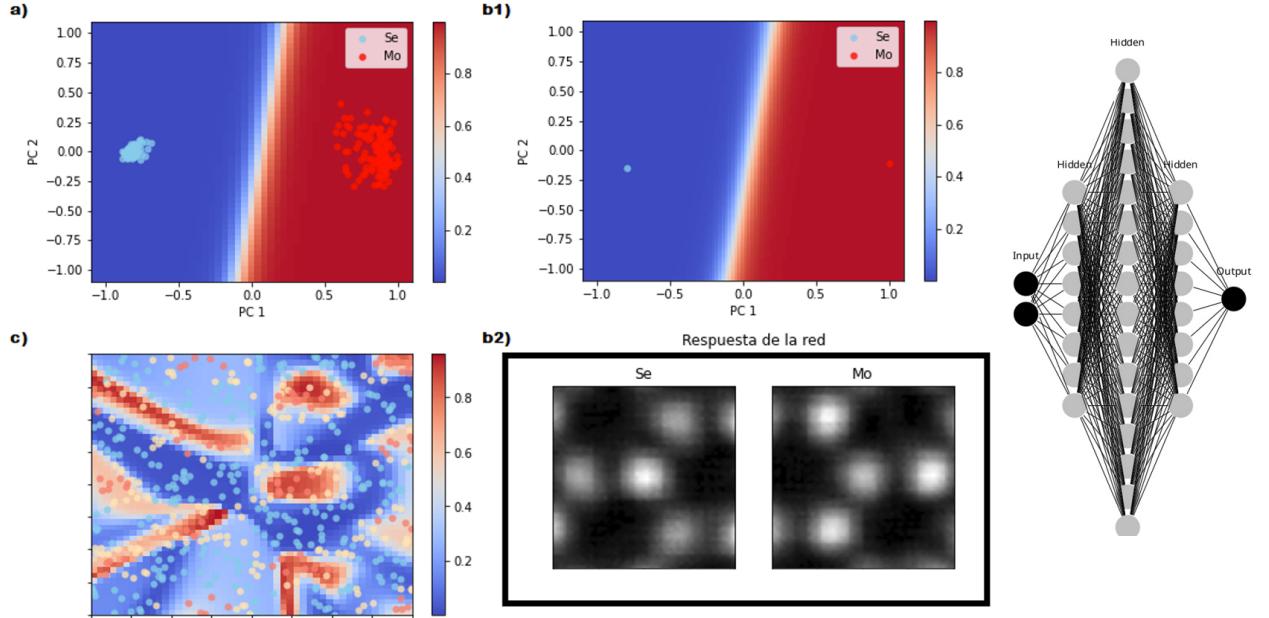
A nivel fundamental, estas redes van a estar compuestas por unas unidades básicas de procesamiento, que llamaremos neuronas, agrupadas por capas. Cada neurona  $n_i^l$  contará con una cantidad de entradas  $x_j^l$  igual al número de neuronas de la capa anterior, un término independiente  $\theta_i^l$  y una función de activación  $f^l$ . Todas las entradas vendrán pesadas por un parámetro  $w_{ij}^l$  que podrá ser modulado durante el entrenamiento (ver [Figura 13](#)).

Cuando la red realiza un *forward pass*, cada neurona de la capa de inicial toma las distintas variables de entrada, las pesa con su parámetro correspondiente y realiza un sumatorio que posteriormente introduce en la función de activación para suministrar una la salida  $y_i^l$ . Cada una de estas salidas actúa como entrada para las neuronas de la siguiente capa, y este proceso se repite hasta alcanzar la capa de salida. Lo que se logra así es realizar una serie de transformaciones lineales encadenadas, llevando los datos de entrada a un nuevo espacio desde el cuál poder clasificarlos (algo similar a lo que hicimos con el PCA).



**Figura 13:** Esquema de una red neuronal artificial. Extraído de la referencia [\[14\]](#).

Para entrenar a la red se realiza un *backward pass*, computando el error  $\delta_i^L$  de la última capa mediante una función de coste (que depende de  $y_i^L$  y  $z_i$ , las salidas que esperamos) y propagándolo hacia atrás para determinar el del resto de capas. Posteriormente se aplica un descenso del gradiente que trata de minimizar cada error  $\delta_i^l$  modulando los parámetros asociados a la capa  $l$ , incluidos los independientes.

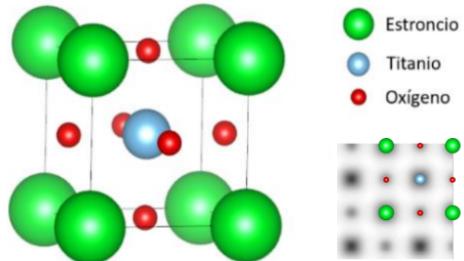


**Figura 14:** Se muestra **a)** el conjunto de entrenamiento (tomados de la 9) y el estado final de la red tras diez mil iteraciones, **b)** una predicción de la red para dos nuevas entradas, y **c)** el caso de una red entrenada con un *dataset* mucho más complejo. A la derecha tenemos un esquema de la topología de la red. Programa desarrollado en Python por el autor [13].

En las Figuras 14a y 14b se recurre a una red neuronal con topología 2-8-16-8-1, de la que se obtienen unos resultados bastante satisfactorios. En el caso concreto de nuestro *dataset*, donde contamos con unos *clusters* perfectamente diferenciados en 2D, realmente no es lo más eficiente recurrir a un algoritmo tan pesado en términos de cálculo. Las técnicas de DL están más enfocadas a situaciones con una complejidad similar, o incluso mucho mayor, a la del caso de la Figura 14c.

Es más, diseñando una red neuronal con la topología adecuada y contando con la potencia de cálculo suficiente, podríamos habernos olvidado de realizar una reducción de dimensionalidad PCA, introduciendo directamente como entrada a la red las observaciones de la matriz  $X$ . En el siguiente capítulo se mostrará algún ejemplo.

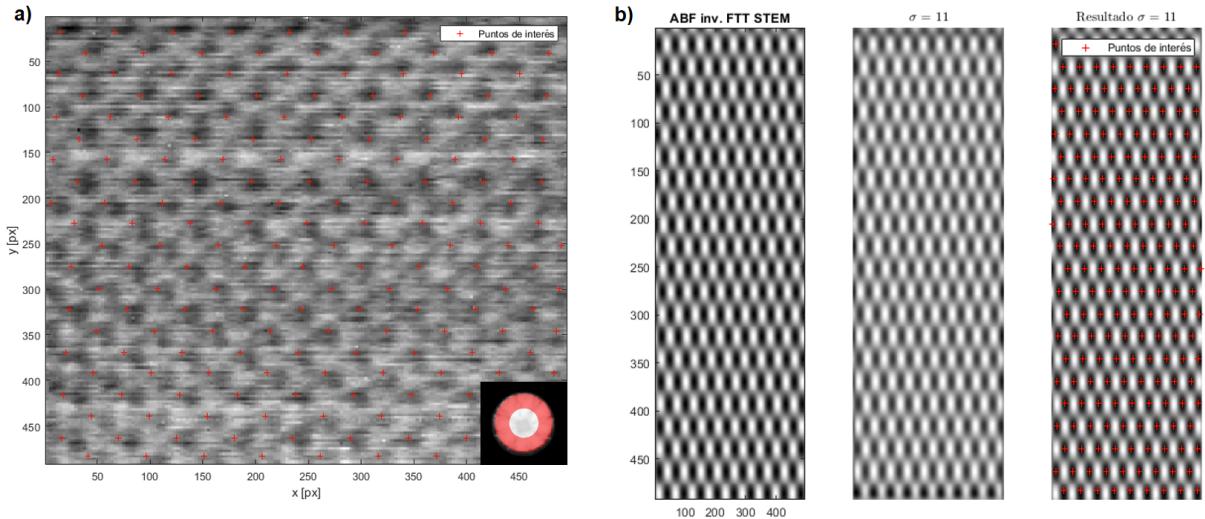
#### 4. Búsqueda de vacantes de oxígeno en STO mediante imágenes 4D-STEM



**Figura 15:** En esta imagen se muestra un esquema de la celda unidad de SrTiO<sub>3</sub> (STO), junto a su visualización sobre una imagen simulada ABF [15].

## 4.1. Procesamiento de imágenes ABF de STO

Partiendo de la imagen ABF que se muestra en la Figura 16, llevaremos a cabo varios procesos de *denoising*. Nuestro objetivo principal es sacar a la luz las columnas de oxígeno que, según se muestra en la simulación de la Figura 15, deberían de poder apreciarse sin problemas sobre la región de integración del patrón de difracción en la que estamos trabajando.



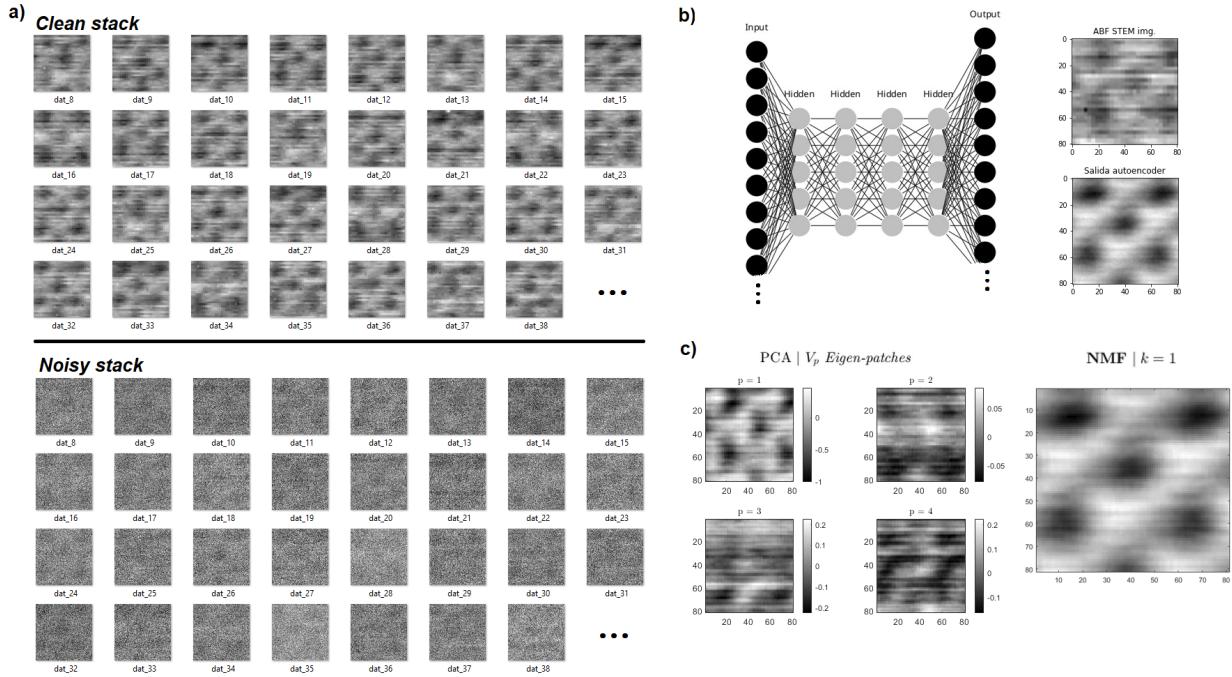
**Figura 16:** a) Imagen ABF tomada por Francisco Fernández Cañizares sobre una muestra de STO horneada a 500 °C durante 8 h - 6C 10cm df0. En la esquina inferior derecha aparece la región de integración sobre el patrón de difracción (15:30 mrads) seleccionada para generar dicha imagen. Los puntos de interés han sido escogidos por el algoritmo de detección de *blobs*, cuyo *input* corresponde con b) un filtrado FFT de alto umbral sobre la imagen ABF invertida. Programa desarrollado en MATLAB por el autor [13].

### 4.1.1. Generación de parches con información sobre la celda unidad

La entrada para los distintos algoritmos de *denoising* será un *stack* de parches (ver Figura 17a) que únicamente van a contener información sobre la celda unidad STO. Para llevar a cabo esta selección, inicialmente aplicaremos un PCA a los parches generados entorno a todos los puntos de interés. Esto nos permitirá filtrar en función de la primera componente principal aquellos parches que corresponden a columnas de Ti.

### 4.1.2. *Denoising* por PCA y NMF

Una vez hemos generado el *clean stack* de celdas unidad, volvemos a aplicar PCA y NMF (Figura 17c), esta vez para realizar un promedio que nos permita extraer la información común entre todas las imágenes del *stack*.



**Figura 17:** Figura que muestra **a)** algunos de los parches contenidos en cada *stack*, **b)** la topología del *autoencoder* junto a su salida tras un entrenamiento de 300 iteraciones a un *learning rate* de 0.2, y **c)** el resultado de aplicar PCA y NMF sobre el *clean stack*. Todo el código escrito por el autor, en relación a este análisis, se encuentra en el capítulo 4 de la referencia [13].

#### 4.1.3. *Denoising* por una red neuronal con topología de *autoencoder*

Un *autoencoder* es un tipo de red neuronal que se va a encargar de codificar las características más críticas de un *dataset* mediante **aprendizaje no supervisado**. La simetría en la topología de este tipo de redes es fundamental, en nuestro caso seleccionaremos  $m^2\text{-}5\text{-}5\text{-}5\text{-}5\text{-}m^2$ , donde  $m^2$  corresponde con el número de píxeles presentes en cada parche.

Si lo que buscamos es utilizar un *autoencoder* para reducir el ruido de una imagen, el primer paso es añadir a nuestro *clean stack* un ruido gaussiano de forma artificial. De este modo, generaremos el *noisy stack* que se muestra en la Figura 17a, el *input* de nuestra red. Una vez contamos con esta entrada, entrenaremos al *autoencoder* imponiendo que su salida sea similar a las imágenes contenidas en el *clean stack*. Así, tras 300 iteraciones, obtenemos el resultado que se muestra en la Figura 17b.

## Referencias

- [1] J.M. Thomas et al., *The rapidly changing face of electron microscopy. Chemical Physics Letters* 631–632 (2015) 103–113.
- [2] Shao and Khursheed, *A Review Paper on “Graphene Field Emission for Electron Microscopy / Appl. Sci.* 2018, 8, 868.
- [3] G. Sánchez Santolino (2015). *Advanced electron microscopy characterization of complex oxide interfaces*. Universidad Complutense de Madrid, pp. 9-30.
- [4] Peter D. Nellist (2011). *The Principles of STEM Imaging*. S.J. Pennycook, P.D. Nellist (Ed.). *Scanning Transmission Electron Microscopy* (pp. 91-92). DOI 10.1007/978-1-4419-7200-2\_2.
- [5] M. Varela et al. (2012). *Scanning transmission electron microscopy of oxides*. Tsymbal, Evgeny Y. et al. (Ed.) *Multifunctional Oxide Heterostructures* (pp. 123-156).
- [6] S. L. Brunton and J. N. Kutz (2017). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*.
- [7] Abruña, Héctor; A Muller, David; Xu, Rui; Ophus, Colin; Miao, Jianwei; Hovden, Robert; et al. (2016): *Nanomaterial datasets to advance tomography in scanning transmission electron microscopy*. figshare. Collection. <https://dx.doi.org/10.6084/m9.figshare.c.2185342>
- [8] N. Naheed et al. *Comparison of SVD and FFT in image compression*. 2015 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 526-530.
- [9] Potapov, P., Lubk, A. *Optimal principal component analysis of STEM XEDS spectrum images*. *Adv Struct Chem Imag* 5, 4 (2019).
- [10] T.J. Atherton, D.J. Kerbyson / *Image and Vision Computing* 17 (1999) 795–803.
- [11] Jiadong D., Xiaoxu Z., Stephen J.P. *A machine perspective of atomic defects in scanning transmission electron microscopy*. InfoMat: Vol 1, No 3, 359-375 (2019).
- [12] Horn, Roger A.; Johnson, Charles R. (1985). "Section 2.6". *Matrix Analysis*. Cambridge University Press. ISBN 978-0-521-38632-6.
- [13] *STEM methods* (2022) Jesús B.V. [https://github.com/jesusBV20/STEM\\_methods](https://github.com/jesusBV20/STEM_methods).
- [14] Arroyo-Hernández, et al. (2013). Sistema de detección y clasificación automática de granos de polen mediante técnicas de procesado digital de imágenes. Uniciencia. 27. 59-73.
- [15] Francisco F. C. (2021). *Advanced microscopy techniques applied to cutting edge material systems*. Universidad Complutense de Madrid.
- [16] K. Szot, W. Speier, R. Carius, U. Zastrow, and W. Beyer. *Localized Metallic Conductivity and Self-Healing during Thermal Reduction of SrTiO<sub>3</sub>*. Physical review letters 88, 075508 (2002).