

# Desarrollo de un algoritmo de conversión de imágenes de mapas y funciones matemáticas a modelos 3D

Estudiantes: Catriel Bartezaghi, Tiago López.

Filiación: Procesamiento Digital de Imágenes 2022.

Tutores: Enrique Marcelo Albornoz, César Martínez.

**Resumen**—El presente trabajo consiste en el desarrollo de una aplicación, que permite obtener modelos 3D a partir de imágenes de mapas o funciones matemáticas, para generar material educativo que facilite la enseñanza a personas ciegas o con discapacidad visual. El algoritmo puede recibir imágenes de mapas políticos o geográficos, y gráficas de funciones matemáticas simples con los colores rojo, verde, azul y negro. Estas imágenes se segmentan por bordes o color, dependiendo el tipo de imagen, y se convierten a un modelo 3D en formato STL.

**Palabras claves**—Segmentación de mapas, imagen a 3D, modelo 3D.

## I. INTRODUCCIÓN

Una problemática presentada desde la ONG<sup>1</sup> Nueva Cultura, abocada a la rehabilitación de personas ciegas y de baja visión, es que a los docentes se les dificulta contar con todos los materiales didácticos necesarios para enseñar a través del tacto. Ejemplos de éstos podrían ser mapas, gráficas de funciones matemáticas o cualquier otra forma que se quisiera enseñar en algún momento. Una solución para este problema sería contar con una aplicación que les permita tomar una fotografía del mapa o función y a partir de la misma obtener una impresión 3D a partir de la generación de archivos digitales compatibles.

En la actualidad se cuenta con programas que podrían tomar una imagen y convertirla a un modelo vectorizado en un formato accesible para un sistema de Diseño Asistido por Computadora (CAD), como lo hacen las aplicaciones Scan2CAD<sup>2</sup>, CADRaster<sup>3</sup> y RaVe<sup>4</sup>. Algunos temas a considerar en este caso, es que la mayoría de estas aplicaciones son de pago y se debería considerar además la conversión del archivo de CAD a un formato que pueda leer una impresora 3D como es STL. Una alternativa posible para resolver el problema se presenta en el siguiente trabajo [1] donde se desarrolla un método para convertir una imagen raster a un formato CAD.

En este trabajo, se propone tomar una fotografía, procesarla a través de técnicas de procesamiento digital de

imágenes (PDI) y luego transformarla directamente a formato STL. En el lenguaje de programación Python podemos encontrar bibliotecas útiles para aplicar estas operaciones como *numpy-stl* [6] que permite convertir matrices de *numpy* a modelos 3D en formato STL. También se encontró un algoritmo de Python *img2stl*<sup>5</sup>, que realiza en forma directa la conversión de una imagen al formato STL. Se investigaron ambas opciones, pero se optó por utilizar la segunda dado que ésta permite realizar la conversión de una imagen a STL de forma simple y directa.

## II. PROPUESTA DE SOLUCIÓN

Para este trabajo se utilizaron como imágenes de entrada mapas que no tengan textos y dibujos hechos a mano de funciones matemáticas. Además, estas funciones y ejes pueden ser dibujados en varios colores (rojo, verde, azul y negro) sobre papel blanco, que luego indicarán diferentes alturas para la impresión 3D. Aunque no se ha probado aún, esto es claramente extrapolable a cualquier tipo de dibujo a mano que respete estas condiciones. Con este tipo de imágenes de entrada se propone el algoritmo presentado en la Figura 1, implementado en el lenguaje de programación Python [3], dentro del entorno de desarrollo de Google Colab [4] y con la biblioteca OpenCV [5], que cuenta con implementaciones de algoritmos de PDI. Al comienzo del algoritmo se lee la imagen y el título de la misma, que se agregará a la imagen final. Luego, se aplica alguno de los distintos procesos de segmentación que se detallan a continuación.

### A. Segmentación de bordes en funciones de un solo color

Cuando la imagen es de un solo color hecha a mano, se aplica el algoritmo de Canny [2] para la segmentación de los bordes con una selección adecuada de parámetros. Los bordes obtenidos tienen poco grosor y al convertir la imagen al modelo 3D se generan picos en la zona superior de los bordes de la imagen. Para evitar esto se aplica un filtro de media aritmética y un umbralizado con umbral 0, con estas operaciones se logra un engrose de bordes en la segmentación. Finalmente, se aplica un cierre morfológico [2] con un elemento estructurante cuadrado de 3x3 para eliminar zonas dentro de la función que hayan quedado en 0 y unir elementos desconectados.

<sup>1</sup> <https://nuevaculturasf.org.ar/>

<sup>2</sup> <https://www.scan2cad.com/>

<sup>3</sup> <https://tessel.com/cadrastert/>

<sup>4</sup> <https://www.raster2vector.com/>

<sup>5</sup> <https://github.com/rmrao/img2stl>

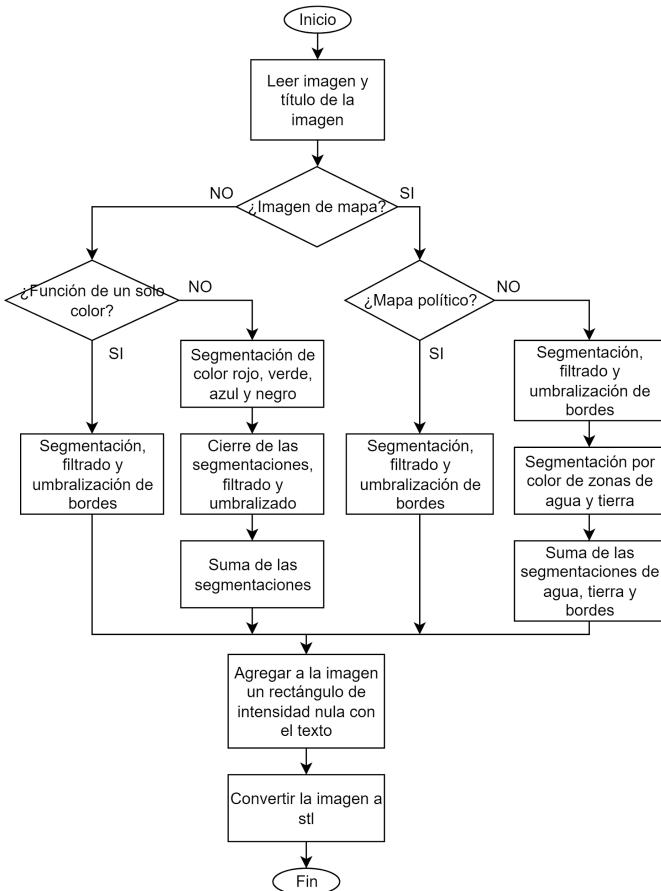


Fig. 1. Diagrama de flujo del algoritmo de conversión de imágenes a modelos 3D.

#### B. Segmentación en funciones a color

Como requisito para las gráficas de funciones en color estas deben limitarse a los colores rojo, verde, azul y negro, sobre papel blanco. Considerando también que los trazos deben ser bien marcados, de otra forma, la segmentación no logrará un buen resultado.

1) *Segmentación en color:* En esta etapa se segmenta la imagen por colores obteniendo 4 segmentaciones (rojo, verde, azul y negro), utilizando la imagen en el espacio HSV [2]. Para los primeros 3 colores se utiliza un rango de valores de S y V que va de 50 a 255, y el H en el rango correspondiente a cada uno de los colores rojo [175,179], verde [50,90] y azul [100,130]. Para el negro se usan S y V de 0 a 50, de bajo brillo y saturación, y un H en el rango de tonos rojos, que fue donde se encontró el tono para el negro. De estas segmentaciones solo utilizamos las máscaras.

2) *Cierre morfológico, filtrado y umbralización:* Las máscaras de la segmentación para cada una de las funciones puede tener cortes o partes desconectadas, por lo tanto, se aplica la operación de cierre morfológico con un elemento estructurante de 5x5 y con dos iteraciones. Luego, se busca agrandar el grosor del borde siguiendo la segmentación anterior, aplicando un filtro de media aritmética y umbralizado

el resultado. De forma que las máscaras queden con valores 0 y 1.

3) *Suma de las segmentaciones:* Se genera la imagen final como una matriz de ceros a la cual se le suman todas las máscaras. Como el objetivo de segmentar por colores es diferenciar los distintos colores en el resultado, se asignan distintas intensidades a cada una de las máscaras. De esta forma, se suman las máscaras negra, roja, verde y azul a la base de ceros, multiplicadas por 50, 60, 70 y 80, respectivamente. Se eligieron esos valores iniciales de intensidad porque generan una diferencia considerable en los bordes y que no es muy distante en el modelo 3D.

#### C. Segmentación en mapas políticos

En primer lugar, se realiza la lectura de la imagen del mapa en escala de grises. Se comienza con una segmentación con el algoritmo de Canny, con los umbrales en 200 y 255 se obtiene una excelente imagen de bordes en todos los mapas sencillos testeados. Luego, se aplica el filtro espacial de desenfoque de 10x10. Esto tiene por finalidad realizar un engrosamiento de bordes. Por último, se binariza la imagen, de manera de terminar de definir explícitamente los bordes. Aquellos píxeles que tengan un valor de intensidad de gris mayor a 10 se consideran bordes y se les asigna un nivel de gris de 255, mientras que los demás se les asigna una intensidad de 0.

#### D. Segmentación en mapas geográficos

El objetivo del proceso, es poder definir un modelo de mapa de relieves de acuerdo a las intensidades de colores presentes en el mapa. Como también se desea obtener los bordes provistos por un mapa político, en el primer paso se realiza la segmentación de la misma manera que se describió en el ítem anterior. Luego, a partir de un mapa de relieve a color, se obtiene la información que determinará las alturas en el modelo impreso, de la siguiente manera.

1) *Segmentación de color:* Se realiza mediante el modelo HSV, ya que éste nos permite definir el rango de matiz en que se encuentran los colores de interés y a segmentar. En este caso, se segmenta el mapa en 2 rangos de colores. Todos los píxeles de la imagen que tengan un valor de matiz (H) dentro del rango [70,110] (rango de 0 a 179 en OpenCV), se consideran como ríos u océanos. Los demás son segmentados como suelo o montaña, esta máscara se obtiene de la resta de la máscara de las zonas de agua. Con esta información se generan dos imágenes aplicando las máscaras a la imagen original.

- 2) *Relieves*: Las imágenes de océanos/ríos y suelo o montaña se convierten en escala de grises, de manera de utilizar las intensidades como información de altura del modelo. En el caso de la imagen de suelo, ocurre que las montañas de mayor altura tienen un menor valor de gris, por lo que la imagen es invertida para que sus valores sean representativos de la altura real.
- 3) *Escalado de relieves*: Para facilitar la distinción de los distintos relieves se realiza una conversión del rango de grises de las distintas imágenes. De esta manera se tienen el agua y la tierra en los rango de [0,60] y [61,200], respectivamente. Mientras que se utilizó 255 para los bordes.
- 4) *Composición imagen final*: La imagen final estará compuesta por distintos valores de las imágenes obtenidas. Debido a que en muchos píxeles hay superposición de valores entre las distintas imágenes, se debe decidir la información de qué imagen es más relevante para el modelo. Por lo que en la asignación de píxeles de la imagen se tienen algunas consideraciones:
  - En caso de coincidencia de imagen de borde con imagen de ríos, se conserva la información de ríos.
  - En caso de coincidencia de imagen de bordes con imagen de suelo, solo se guarda la información de borde.
  - En los demás casos no hay coincidencia por lo que se asigna el píxel que corresponde.

En esta etapa del algoritmo ya contamos con la información de la imagen que será impresa en 3D, se agrega el texto en un rectángulo de ceros insertado en la parte superior de la imagen a imprimir. A partir de aquí tenemos la imagen final, se convierte la misma al formato STL a partir del algoritmo *img2stl* y se devuelve el modelo 3D.

### III. RESULTADOS

En esta sección se presentan los resultados obtenidos de la aplicación del algoritmo, comenzando por el mapa político [7] de la provincia de Santa Fe. Se aplica el procesamiento indicado para mapas políticos, en la Figura 2 se observa el modelo 3D obtenido al pasar la imagen por el algoritmo. Se puede notar la suavidad de los bordes y que están bien marcados. En la Figura 3 se presenta una vista del modelo 3D del mismo mapa pero en este caso se aplicó un filtrado con una máscara más pequeña, lo que resulta en bordes más finos. Se puede observar que en los bordes que deberían ser suaves hay picos de altura variable. Para solucionar esto se aumentó la talla del filtro, el resultado se presenta en la Figura 4, en este caso los bordes son suaves y tienen todos la misma altura.

Una vez obtenido el modelo 3D lo enviamos a impresión, el resultado se presenta en la Figura 5. Todos los bordes del modelos están bien delimitados y la sección superior de los bordes es suave.

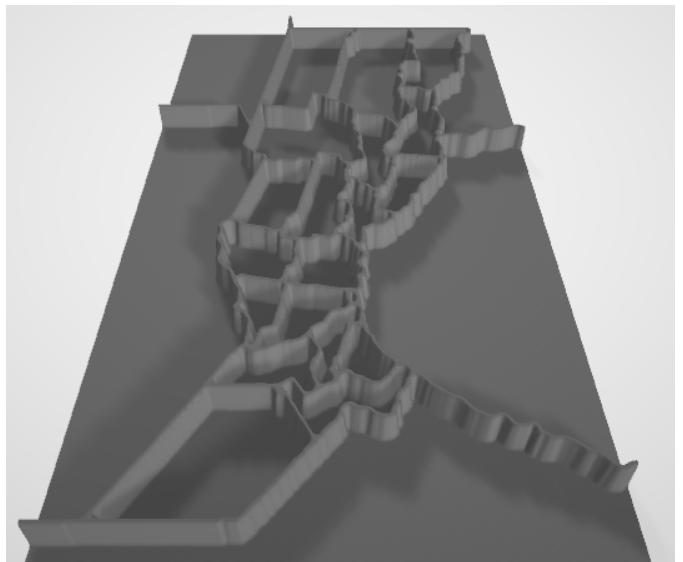


Fig. 2. Modelo 3D del mapa de la provincia de Santa Fe.

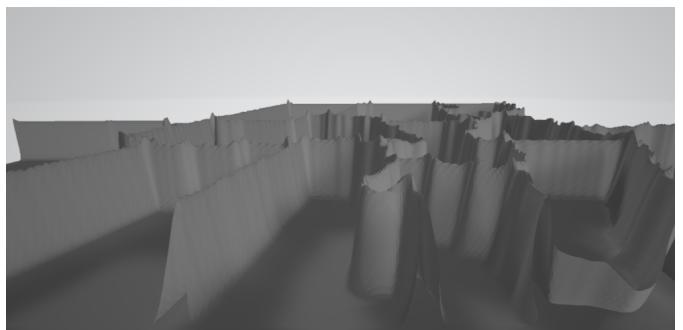


Fig. 3. Visualización de los picos en el modelo 3D.

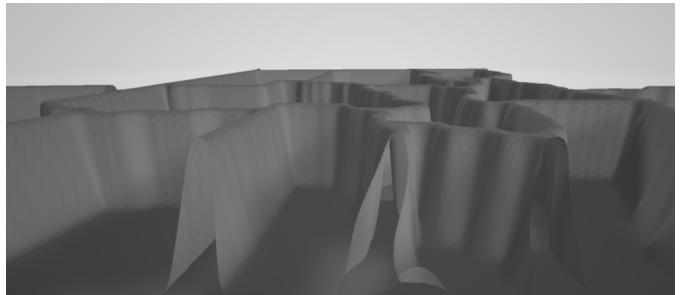


Fig. 4. Modelo 3D sin picos.

También se imprimió un modelo 3D de una imagen de funciones con distintos colores que se graficaron a mano, presente en la Figura 6. En esta impresión se pueden notar las diferentes alturas de las funciones, que dependen de los distintos colores de la entrada. Este modelo también tiene añadido el título de la impresión, que es suave y claro para la lectura.



Fig. 5. Modelo 3D impreso de la provincia de Santa Fe.

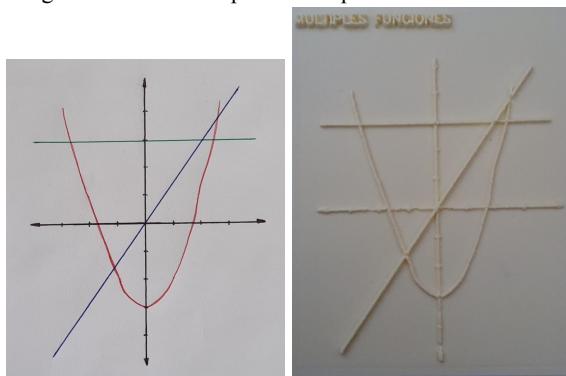


Fig. 6. Conversión de imagen de funciones en colores al modelo 3D.

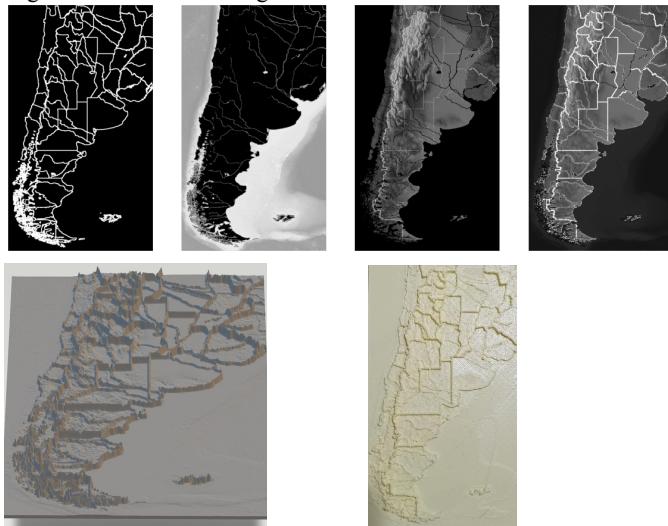


Fig. 7. Segmentación de mapa de relieves, modelo 3D e impresión final.

Por último se muestran los resultados del mapa de relieves [8,9]. Los resultados fueron muy buenos, conservando la mayoría de detalles. El principal inconveniente es que debido a la precisión de la impresora, se pierden detalles finos como son los ríos. En la Figura 7 se puede observar un resumen en

imágenes del proceso de segmentación, el modelo STL generado y por último el resultado obtenido con una impresora de filamentos utilizando material PLA de 1.75mm.

#### IV. CONCLUSIONES

En este trabajo se presentó un algoritmo para obtener modelos 3D a partir de imágenes de mapas y gráficas de funciones matemáticas. Restringir el tipo de entrada al algoritmo permitió reducir el preprocesamiento de las imágenes y reducir la complejidad del algoritmo. De todas formas se pueden obtener modelos 3D detallados, con información espacial relevante de los mapas y las funciones.

En todos los casos el aplicar un filtrado con una máscara mayor nos permitió añadir grosor a los bordes y solucionar el inconveniente de los picos generados en la parte superior de los bordes.

Como trabajo a futuro en el corto plazo se puede considerar dar un tratamiento a distintas características de los mapas que pueden afectar negativamente la impresión, como lo son: texto, puntos de ciudades, líneas. También se podría extender el método utilizado con funciones matemáticas a imágenes generales de trazos e incorporar la conversión a braille del texto.

En el mediano plazo se puede desarrollar una algoritmo en el que los colores sean representados por distintos patrones, de manera que estos sean aplicados en la zonas color homogéneas y poder ser reconocidos con el tacto. A largo plazo con este mecanismo patrón-color se podrían desarrollar infinitades de modelos a partir de imágenes. Finalmente, implementar una aplicación móvil con el algoritmo.

#### AGRADECIMIENTOS

Al Ingeniero César Arrasin y a Juan Mauro Loreficcio por responder a las consultas técnicas y realizar las impresiones 3D.

#### REFERENCIAS

- [1] Intwala, A. (2019). Image to CAD: Feature Extraction and Translation of Raster Image of CAD Drawing to DXF CAD Format. CVIP.
- [2] Gonzalez, R. y Woods, R. (2008) Digital Image Processing. 3rd Edition, Pearson Prentice Hall, Upper Saddle River.
- [3] Van Rossum, G. y Drake, F. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.
- [4] Bisong, E. (2019). Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA.
- [5] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.
- [6] Harris, C., Millman, K., van der Walt, S., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. (2020). Array programming with NumPy. Nature
- [7] Milenioscuro (2017). *Argentina Santa Fe location map* [Imagen]. Wikipedia.  
[https://es.wikipedia.org/wiki/Archivo:Argentina\\_Santa\\_Fe\\_location\\_map.svg](https://es.wikipedia.org/wiki/Archivo:Argentina_Santa_Fe_location_map.svg)
- [8] NordNordWest (2022). *Argentina location map* [Imagen]. Wikipedia.  
[https://es.m.wikipedia.org/wiki/Archivo:Argentina\\_location\\_map.svg](https://es.m.wikipedia.org/wiki/Archivo:Argentina_location_map.svg)
- [9] NordNordWest (2018). *Relief Map of Argentina* [Imagen]. Wikipedia.  
[https://es.m.wikipedia.org/wiki/Archivo:Argentina\\_location\\_map.svg](https://es.m.wikipedia.org/wiki/Archivo:Argentina_location_map.svg)